The Analysis, Modeling and Detection of Botnet-based Hosting Services and Emerging Threats

by

Matthew S. Knysz

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2012

Doctoral Committee:

Professor Kang G. Shin, Chair
Assistant Research Scientist Michael D. Bailey
Assistant Professor J. Alex Halderman
Assistant Professor Qiaozhu Mei

To my parents, Rick and Janet, and my brother, Adam, with love and thanks.

ACKNOWLEDGEMENTS

First and foremost, I would like to offer my sincere and deepest gratitude to my adviser, Professor Kang G. Shin, without whose patience, inspiration, encouragement and guidance this would never have been possible. He has been an exemplary adviser, providing invaluable support and advice both towards my academic career and life in general. I will forever remember his many words of wisdom, and I count myself truly blessed and fortunate to have been his student and studied under him all these years. I would also like to thank Professor J. Alex Halderman, Dr. Michael Bailey and Professor Qiaozhu Mei for serving on my dissertation committee and providing such valuable insight and feedback towards improving this thesis. Thanks to the past and present members of the Real-Time Computing Laboratory (RTCL) for providing such a friendly and welcoming environment for my graduate studies. I will always look back fondly on our many outings and social events. I would like to especially thank those in the security, networking and other collaborative groups for all their technical insights and help towards my graduate work, including Taejoon Park, Abhijit Bose, Zhigang Chen, Min-Gyu Cho, Hyoil Kim, Alexander Min, Jisoo Yang, Xiaoen Ju, Xinyu Zhang, Ashwini Kumar, Eugene Chai, Caoxie Zhang, Pradeep Padala, Yu Wu, Kyu-Han Kim, Karen Hou, Fangjian Jin, Antonio Kim Chang-Hao Tsai, Kassem Fawaz and many others. I would like to give a special thanks to my colleague Xin Hu, with whom I spent countless hours brainstorming, working on papers, and making fond memories at conferences around the world; a better coauthor, fellow researcher and friend would be hard to find. Thanks to Arun Ganesan for all his hard work

and dedication to the final chapter of this thesis and to Katharine Chang, Yuanyuan Zeng and Hahnsang Kim for their help and friendship over the years. I am grateful to Stephen Reger, the RTCL Secretary, for his administrative help and friendship during my time in Ann Arbor. I would also like to thank Laura Fink for all her technical help and supportive companionship. I was also fortunate enough to intern at Bell Laboratories, and would like to thank my mentors there, Dr. Li Erran Li and Dr. Jin Cao, for their support and guidance. A very special thanks to my parents, Rick and Janet, whose love and support have shaped me into the man I am today and inspired me to always strive for my goals, whatever they might entail. Finally, the work reported in this dissertation was supported in part by the Office of Naval Research under Grant N00014-09-11042.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

The Analysis, Modeling and Detection of Botnet-based Hosting Services and Emerging Threats

by

Matthew S. Knysz

Chair: Kang G. Shin

Botnets—vast collections of compromised computers (i.e., bots) under the control of a botmaster—have become one of the greater threats facing the Internet community due to their versatility and financial appeal. Much of their success, financial and otherwise, can be attributed to 4 properties/strategies: *stealth*—first and foremost, bots want to remain stealthy in their infection and occupation, keeping botnet resources high; *modularity*, granting bots new functionality by allowing already infected machines to update their bot malware; *Command and Control*, permitting coordination and post-deployment modification of the botnet functionality and behavior as needed for various scams or to evade detection; and *content-delivery mechanisms*, such as botnet-based hosting services and FF DNS-advertisement strategies, permit botmasters to serve scams and malicious content to victims for profit or the purpose or swelling their botnet ranks.

The dissertation addresses this stealthy aspect of botnets and its imposed limitations, exploring botnets' primary content delivery mechanism—botnet-based hosting services utilizing FF DNS-advertisement strategies—and the future mobile botnet

threatscape emerging with the increase in mobile devices and wireless connectivity. It introduces and evaluates an automated enterprise solution, called RB-Seeker, for accurately detecting domains and bots involved in botnet-based hosting services. It grants insight into the global DNS-advertisement strategies and limitations FF botnets by deploying DIGGER—a distributed DNS-monitoring system comprising hundreds of nodes spanning multiple continents—for an extended period of time, identifying intrinsic behavioral-detection features and evaluating if current botnet resources are sufficient to mimic benign domains and evade detection. Finally, using real-world WiFi network locations, mobility traces and bus routes for the city of San Francisco, it simulates highly mobile botnets utilizing only open WiFi networks, demonstrating that they can pose a serious threat and provide an ideal mechanism for botmasters transitioning to the mobile landscape. This dissertation demonstrates that the powerful distributed systems granted by botnets can support numerous stealthy evasion tactics, requiring a more intimate knowledge of botnet resources and capabilities so that properties intrinsic to their functionality can be more effectively targeted and exploited. It gives valuable insight into these intrinsic properties and resource limitations of both current and future botnets, providing more resilient detection and disruption approaches.

CHAPTER I

Introduction

1.1 Background

As the Internet and computer systems have become more staple and ubiquitous components of our everyday lives, attacks against them have experienced a fundamental paradigm shift, evolving from isolated, destructive, proof-of-concept attacks carried out primarily for 'bragging-rights' to a more systematic and organized endeavor aimed at generating profits. Perhaps the best example embodying this new profit-based mentality is the emergence and constant growth of botnets: systems containing thousands of compromised computers under the command of a single botmaster. Botnets adopt a novel, hybrid approach when it comes to malware, incorporating various aspects of its predecessors (i.e., trojans, viruses, worms) into its bot malware in a way that promotes profits. Like trojans, bots install backdoors on their compromised systems, allowing attackers to issue remote commands. However, unlike trojans, which typically grant attackers control to a single system at a time, bots introduce a Command and Control (C&C) channel, permitting simplified control of the entire distributed systems of bots and resulting in an unprecedented level of free computing power. Bots make use of propagation mechanisms traditionally attributed to viruses (i.e., spreading by attaching themselves to infected files) and worms (i.e., spreading through network vulnerabilities) as well as all new mechanisms, such as

leveraging social connections and drive-by scripting attacks. Unlike viruses, bots do not typically attempt to cause damage to the underlying system once compromised; doing so would hinder the usefulness of the newly acquired bots and result in them being more easily detected and shutdown. Likewise, bots do not attempt to spread as rapidly and without bounds as worms since this, too, would cause their activity to be more easily detected, crippling the botnet's potential power. On the contrary, bots attempt to spread more slowly and discretely, often patching up vulnerabilities after gaining access to a system to prevent competing botmasters from gaining control of their valuable resources. It is far more lucrative for a botmaster to retain control of a sizable botnet than to spread rapidly (worms) and cause noticeable damage (viruses). This is because the value of a botnet lies in the power such a massively connected system of compromised computers can afford. Like shadow governments, botmasters have realized that their power can be wielded longer by staying out of the spotlight, making stealthy operations of paramount importance to a successfully functioning and lucrative botnet. This stealthy property/strategy of botnets extends throughout bots' entire life cycles, from their propagation and infection to their operations. Since building a massive and powerful botnet in a stealthy manner takes time and resources, botmasters also stealthily execute their scams in order to retain their compromised constituent bots for as long as possible; stealthy scams are also more likely to remain undetected and operational longer, increasing their potential for profits.

A further distinction of bots over earlier malware is their use of updatable and modular code. When combined with their versatile C&C channel, this allows botmasters to quickly and easily update their bots with new functionality or evasion strategies as needed. In this way, botnets resemble powerful distributed computing systems running a suite of upgradable money-making software. This makes botnets a versatile computing platform capable of running numerous, simultaneous scams, including Distributed Denial of Service attacks (DDoS), information theft (e.g., credit

cards, passwords, entire identities), hosting services, sending spam, and phishing campaigns. Furthermore, by using encrypted communication in their C&C channels and upgrading binaries to evade the currently dominate signature-based detection strategies of most Anti-Virus (AV) systems, bots are notoriously difficult to detect, making them a scourge to the security community. Even newer behavior-based detection strategies can be evaded by botnets if the monitored behavior isn't intrinsic to the botnet's functionality; a simple bot upgrade with a slightly different behavioral approach can render such detection strategies impotent. This phenomena has already been observed in the wild. For instance, when behavioral detection strategies began targeting botnets' once-popular IRC-based C&C mechanisms, botmasters responded by incorporating encrypted channels or switching to HTML- or P2P-based approaches.

The potential for profits that botnets provide has spawned a new underground economy, which, in turn, has resulted in a proliferation of "off-the-shelf" malware toolkits for the automatic generation of generic bots. These toolkits, which can be purchased by aspiring botmasters, allow even novice programmers to quickly develop and deploy new bot variants. The newly acquired botnets can then be sold to interested buyers, either in their entirety or as a particular service, such as sending spam email. More than anything, it is this shift in mentality, from malware being a source of destruction and prestige to a viable model for profits, that has necessitated the adoption of stealthy approaches and propelled botnets to a major contender in the Internet threatscape. The influx of money into the malware scene has resulted in not only an explosion of new malware variants, but also a class of much more stealthy and dangerous malware, endued with the power and resources typically attainable only by corporations and governments with powerful distributed computing systems. Botnets' success, financial and otherwise, can be attributed to four major properties/strategies. First and foremost, in order to keep their resources high enough to

support their lucrative scams, bots must remain *stealthy* during their infection and occupation. To do otherwise would result in their detection and mitigation, severely decreasing the power and versatility gained from such a distributed system and reducing the effectiveness of their scams and potential to generate profits. Second, to help maintain their stealth and keep botnet activities adaptable, bots must be *modular* in nature, allowing them to be easily updated post deployment by adding new functionality and evasion techniques. Third, to permit their coordination for scams and their post-deployment modification of functionality, botnets require a *Command and Control* channel, granting botmasters the control necessary to alter their activities and behavior to evade detection and target new victims. Lastly, to perpetrate their scams and infect new victims, botnets depend upon *content-delivery mechanisms*, such as botnet-based hosting services and FF DNS-advertisement strategies. Throughout the dissertation, we study this stealthy aspect of botnets and its imposed limitations, exploring botnets' primary content-delivery mechanism—botnet-based hosting services—and the future mobile botnet threatscape emerging with the increase in mobile devices and wireless connectivity.

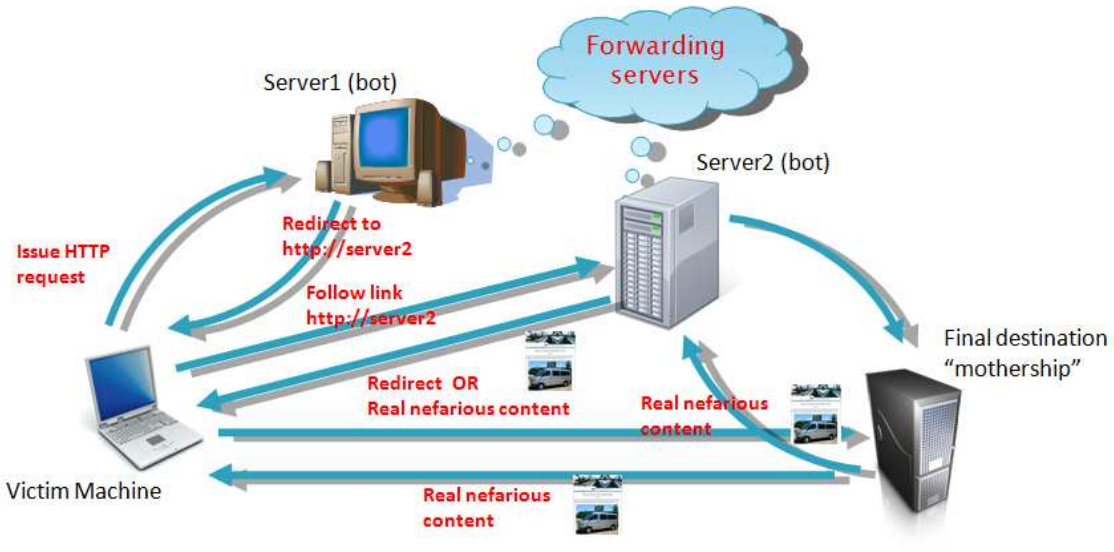


Figure 1.1: Fast-Flux botnet redirection/proxy mechanism

1.2 Motivation

Due to their tremendous success in generating profits, the botnet threat is likely to increase and become even more popular in the future. This dissertation is motivated by the following observations concerning botnets:

1. Stealth is intrinsic to botnet success. If not stealthy during propagation, infection and occupation, bots can be more easily detected and removed, decreasing the botnet's overall resources and its ability to perpetrate scams and attacks. Likewise, if the scams and attacks themselves are not operated in a stealthy manner, they are more easily discovered and quickly shutdown or mitigated, rendering them less effective and resulting in a loss of profits. Making use of this relationship, we will explore ways to make stealthy operations more difficult for botmasters, easing their detection while reducing their potential power and making their scams less effective.

2. Since botnets typically host a variety of nefarious scams, the detection and removal of bots involved in one particular scam can potentially hinder the entire botnet, decreasing its overall resources and indirectly hurting the other scams it perpetrates.

3. Many scams rely on victims visiting malicious websites for the purpose of infection, phishing for personal information, or hawking wares. Botnets provide an ideal environment for anonymously hosting such malicious content, shown in Figure 1.1. Victim machines are redirected to multiple bots before finally reaching the actual content server; alternatively, or in conjunction with redirection, the content can be proxied to the victim through an intermediary bot or actually hosted on a bot—though distributing content to be hosted on so many different bots is much more difficult to manage than a centralized approach. In

this way, botmasters obtain an additional layer of misdirection and protection for themselves and their nefarious content.

4. In order to successfully serve as a hosting service, botmasters must advertise their bot IPs using the publicly available Domain Name Service (DNS), allowing new victims to locate their bot-hosted content. Consequently, there is essentially a public record of active bot IPs, provided botnet domains can be identified based on their DNS-advertisement behavior.

5. The bots composing a botnet consist of a variety of heterogeneous compromised computers, all with varying local resources and network connectivity. Individual bots may go offline at any moment for numerous reasons, including the compromised computer being shutdown or having the bot malware detected and removed. Consequently, botnet-based hosting services must account for this unreliable nature by adopting an unconventional approach to DNS advertisement, earning them the moniker of Fast-Flux (FF) domains.

6. Mobile devices are rapidly becoming more powerful, prolific, and the preferred access point of many for their online activities. With their expanding application markets, standardized OSes, and advancements in processing power and memory, they are quickly approaching the capabilities of modern computers. However, their multiple communication channels (i.e., WiFi, 3G/4G, Bluetooth, SMS and MMS messaging) and always-on connectivity allow for more advanced and stealthy attacks than are possible with traditional computers. Combined with their high mobility, mobile devices can be leveraged by botmasters to attain an unprecedented level of stealth, permitting longer-lasting and more lucrative scams. Consequently, they provide an attractive target for botmasters, and it is only a matter of time before mobile botnets emerge in force on the Internet threatscape. A preemptive analysis of the covert techniques made possible with

mobile bots and their limitations can provide valuable insight in how to curtail their nefarious impact.

From these observations, a potential strategy for combating the botnet threat takes shape. With stealth such an important contributor to botnets' long-term success and financial gain, uncovering and exploiting botnets' limitations for functioning covertly can provide powerful improvements in detection and mitigation. One such scam, botnet-based hosting services, is particularly well-suited to this approach since the DNS records necessary for its functionality are publicly available and easily monitored. This makes stealthy operations more difficult for botmasters since it prevents the use of encryption or covert channels, facilitating behavior-based detection. Furthermore, since botnets are used for numerous different scams simultaneously, detecting and shutting down bots involved in one scam can have a significant impact on their other scams, requiring them to give up this particular scam entirely or risk possible detection and disruption. Therefore, a primary focus of this dissertation is to better understand the stealthy limitations of botnets with respect to the DNS advertisement of botnet-based hosting services and the burgeoning mobile landscape, permitting the development of better detection, mitigation and disruption strategies.

1.3 Research Goals

Motivated by the financial success of botnets and their necessary use of stealth, this dissertation focuses on a better understanding of the current and future stealthy capabilities and strategies of botnets for perpetrating their nefarious scams (in particular, botnet-based hosting services and mobile botnets), allowing for more resilient approaches in detection and mitigation and the development of a more robust and accurate detection system. It has the following objectives:

- We aim to develop a real-time detection system capable of fast and accurate de-

tection of botnet-based hosting services targeting an enterprise network. To do so, we will attempt to make use of multiple input sources to identify suspicious domains and then identify those belonging to botnets based on their publicly available DNS records.

- We attempt to better understand the various differentiating features distinguishing FF botnet domains from other domain types in terms of their DNS-advertisement strategy. To do so, we will monitor the DNS behavior for different domains from a unique *global* perspective, giving us a better understanding of botnets' overall content-hosting resources.

- Based on extensively gathered DNS data, we hope to model the current resources available to FF botnets. This will allow us to determine if DNS monitoring can ultimately serve as a viable option for botnet detection or if the versatility provided by botnets is sufficient enough to mimic benign domains' DNS-advertisement behavior and evade detection.

- Using our newfound understanding of FF botnet DNS-advertisement strategies and their capability for mimicry attacks, we will propose a more powerful detection approach. Like the previous enterprise-level solution, it should be able to do so quickly and accurately. We hope that our new insight into current botnet resources will allow for the creation of a detection mechanism that is more resilient or immune to possible mimicry attacks.

- Lastly, we will look to the future of botnets and explore their viability on mobile devices using open WiFi networks for communication and perpetrating scams. Such an approach, if achievable, would allow botnets to quickly begin harnessing the newly emerging mobile market by adding mobile devices to their already swollen botnet ranks.

1.4 Research Contributions

The work in this dissertation has demonstrated the vast power and versatility distributed systems of compromised computers and mobile devices can obtain with proper management. Botnet-based hosting services, with their FF DNS-advertisement techniques, can achieve a content-distribution system comparable to industry-level Content Distribution Networks (CDNs). Through clever manipulation of their resources, they can even appear benign to existing detection systems, allowing their scams to continue unabated for longer durations. Likewise, through the cunning use of open WiFi networks, botmasters can turn an otherwise chaotic swarm of highly mobile, low-end devices into an efficacious and stealthy attack system. This capacious adaptability is attained through the organized cooperation of numerous and diverse compromised components. Should the distributed systems' currently swollen ranks decrease, much of this flexibility would be lost, severely decreasing botnet capabilities and their potential for profits. Consequently, we find stealth to be integral to continued botnet operations, lest too many bots are discovered and removed from service. However, our work has shown that even an unreliable distributed system of compromised computers can, if large enough, adapt to new detection approaches efficiently and effectively with high statistical probability. Thus, advances in detection are quickly met with advances in evasion, ultimately relying on intrinsic limitations of the distributed system's underlying components—whether they are the unreliable connectivity and diurnal distribution patterns of compromised computers or the rapid, vehicular mobility of hand-held devices—to disrupt stealthy operations and expose botnet activities. Certainly, security is an ever-escalating arms race where, in this digital age, the stakes are drastically increasing. Faced with the raw power and versatility afforded by massive distributed systems of bots, which are soon to be joined by mobile and always-on bots, it is increasingly important for security professionals to understand the underlying nature and limitations of the apparent threat if there

is to be any hope of disruption. This dissertation has strived to further this understanding, both for the current Internet threatscape and for that which looms on the horizon, presenting detailed and practical approaches towards botnet detection and disruption. Specifically, it has made the following research contributions:

1.4.1 Real-Time Botnet Detection for Enterprise Networks

We have designed and implemented a prototype system called Redirection Botnet Seeker (RB-Seeker) for detecting botnet-based hosting services utilizing redirection. RB-Seeker achieves fast and automatic detection of botnets, irrespective of their C&C protocol or structure, by employing several statistical correlation and classification techniques to analyze network traffic and DNS behavior. The system begins with two parallel subsystems, the SSS and the NAS, as first-line filters, cooperatively detecting redirection domains from multiple data sources. The SSS inspects spam emails for embedded links participating in redirection, while the NAS explores unique temporal/spatial features (e.g., inter-flow duration, flow size) of typical redirection activities so that redirection domains can be identified without the expensive inspection of packet payloads. The second-line detector, the a-DADS, exploits the atypical DNS-query statistics of Redirection Botnets (RBnets) to distinguish between malicious and legitimate domains. Our evaluation of RB-Seeker on real-world traces showed it's capable of detecting both Aggressive and Stealthy RBnets with few false positives. Because of the prevalent and major role botnets play in redirection infrastructures, fast and automatic detection of such RBnets can not only protect users from phishing and scam websites, but also potentially deter many other malicious activities commonly perpetrated by botnets. Furthermore, RB-Seeker is expected to be incrementally deployable and easily incorporated into existing security systems since its data sources are readily available in most enterprise networks.

1.4.2 Understanding of FF Botnet Global DNS Behavior

We have examined the global IP-usage patterns exhibited by different types of malicious and benign domains, including single FF (FFx1) and double FF (FFx2) domains. We have deployed DIGGER, a lightweight DNS-probing engine, on 240 PlanetLab nodes spanning 4 continents. Collecting DNS data for over 3.5 months on a plethora of domains, our global vantage point enabled the identification of various IP-usage patterns inherent to the operations of the different domain types. Conducting a detailed analysis, we were able to determine distinguishing behavioral features between the domain types based on their DNS-query results. We have quantified these features and demonstrated their effectiveness for differentiation by building a multi-leveled, multi-week SVM classifier capable of discriminating between five domain types: CDN, non-CDN/MAL, FFx2, FFx1_Arec and FFx1_NArec. Applying our classifier on a set of 5,171 unknown domains produced promising results, correctly categorizing the domains with only 1 false positive—due to a CDN exclusively using a load-balancing DNS-advertisement strategy. Our classification results showed the relative distribution of the domain types in our testing data and the current state of FF domains, including the increased presence and versatile implementation range of FFx2 domains. We have shown that fluxiness is typically more pronounced in A recs and that there is an apparent trend towards using FFx1_NArec domains, which were previously unseen in the wild.

1.4.3 Mimicry Attack Modeling/Analysis and Improved Detector

We have examined the current, state-of-art, DNS-based FF detectors, analyzing their effectiveness in detection. In doing so, we developed accurate models for bot decay, online availability, DNS advertisement, and performance, which we used to evaluate novel mimicry attacks against these detection systems. Based on these models, empirical evidence, and logical assumptions, we have demonstrated that cur-

rent botnet resources are sufficiently capable of subverting state-of-art FF detection mechanisms. We have discovered evidence of current FF domains adopting aspects of our proposed mimicry attacks, although they aren't managed as assiduously as our optimal models assume and, therefore, still possess room for improvement. We have shown that the incorporation of more advanced views—such as an extended detection window—serves to encumber mimicry attacks by introducing additional, necessary parameters unknown a priori to botmasters. We introduced a novel spatial-detection system utilizing 5 cooperating monitoring nodes on different continents, forcing botnet mimicry attacks to replicate the location-aware DNS-advertisement strategy of CDNs to evade detection. We modeled such a system and demonstrated that, as with previous detection systems, current botnet resources can successfully evade detection at the expense of online availability and performance. We found that current botnet resources are sufficient for mimicry attacks against increasingly advanced detection systems, but that such attacks decrease their connectivity and performance. Taking advantage of this relationship, we introduced a new detection metric for defending against mimicry attacks, percent connectivity, that measures the percentage of online IPs advertised. Using our models, we showed that percent connectivity can make existing detection systems more resilient to mimicry attacks, requiring botmasters to increase their diligence with shorter TTLs to succeed. When combined with our proposed spatial-detection system, we found that even the largest botnets currently lack the resources necessary to evade detection continuously within a 24-hour period.

1.4.4 Analysis of Future WiFi Botnet Threat

We leverage real-life cab mobility traces, bus routes and actual open WiFi AP locations to successfully simulate the C&C, DDoS attacks and spam attacks of a mobile botnet using only open and unencrypted WiFi networks. We have shown that such a mobile botnet, traveling quickly through an urban environment in vehicles, can

successfully achieve an HTTP-based C&C channel with a fine level of control, with new commands typically propagating to more than 75% of the cab botnet within 2 hours of injection—sometimes, within as little as 30 minutes of injection. For our bus botnet, we found that, within a single hour, botmasters can reach over 80% of their total potential botnet, often reaching more than 90%. Moreover, those bots able to receive commands usually have an ≈30–50% probability of being able to do so within a minute of the command being issued. With our cab botnet, we have shown that even a small mobile botnet of 536 devices can successfully mount a DDoS attack against unprotected systems or systems using default firewall settings. Utilizing intelligent AP selection, our bus botnet has demonstrated that mobile WiFi botnets can potentially mount even more powerful DDoS attacks, typically capable of over 2 million SYN packets per hour (≈555 per second) and as many as 4 million (≈1,100 per second) during its peak hour. We have also established that a highly mobile botnet can serve as a powerful spamming mechanism, despite being limited to only open and unencrypted WiFi APs. Our small cab botnet was able to issue ≈20,000–300,000 spam emails hourly during peak weekday and weekend hours and between ≈400,000 and 6.4 million emails daily during weekends; even when issuing spam during only weekday rush hours (i.e., 8 hours a day) it was able to send between ≈150,000 and 2.3 million emails daily. We discovered similar results for our bus botnet, which could issue between ≈600,000 and 11.7 million spam emails daily and over 1 million per hour in certain instances. Furthermore, simulations have demonstrated that the C&C, DDoS attack and spam attack traffic were all adequately distributed across open WiFi networks, and no single open AP was over utilized at any given moment; together, these results affirm the stealthy nature of mobile WiFi botnets, making them especially alluring to botmasters. Finally, we provided some simple defensive mechanisms, showing their overall effectiveness in disrupting botnet operations.

1.4.5 Related Work

Botnets have now become one of the biggest threats to the Internet community. Most of the previous research focused on analyzing and understanding the operations and threats of botnets [41]. Cook *et al.* [28] studied the structure of botnets and highlighted the potential threats of peer-to-peer (P2P) botnets. They also showed that detecting botnets based solely on the C&C channel is not effective. Rajab *et al.* [62] constructed a distributed measurement infrastructure to measure botnets' Internet Relay Chat (IRC) activities and found that botnets contribute the majority of unwanted traffic in the Internet. Botnets' diurnal properties are studied in [30] and used to model the propagation of botnets. More recently, P2P botnets appeared in the wild that use the P2P infrastructure as the C&C channel and, therefore, are more robust against node failures and difficult to be taken down. Grizzard *et al.* [33] analyzed the architecture and communication protocol of the P2P botnet Peacom (a.k.a. *storm worm*) [23]. A model for advanced hybrid P2P botnets has also been proposed in [71], which provides robust connectivity, control traffic dispersion, encryption, easy recovery and many other techniques that significantly improve the capability of P2P botnets in surviving the node failure and shutdown of C&C channels. Most of these methods fall into the category of passive analysis. To gain an insider view of a botnet, researchers also took more active approaches, infiltrating botnets with actual malware samples or customized crawlers. For example, Holz *et al.* [40] crafted a specific P2P client to join the Storm Worm's P2P botnet and estimate the total number of compromised machines. Researchers also disrupted the Conficker botnet by sinkholing future DNS domains of the C&C server, preventing botmasters from updating the infected hosts [42]. More recently, Stone-Gross *et al.* [66] successfully took over the Torpig botnet for ten days by preemptively registering DNS domains the bots would be contacting as C&C servers in the near future. This allowed them to reveal detailed operations of the Torpig botnet and accurately estimate the number of compromised

hosts.

While there are numerous strategies to mitigate the effects of malware, most of them are ill-suited for combating botnets. Due to the modular nature of bots and the popularity of bot-development toolkits, it becomes fairly easy for even moderately skilled hackers to acquire botnets of their own, churning out numerous bot variants before up-to-date signatures can be generated for a signature-based anti-virus or intrusion detection system. In addition, sandboxes, honeypots, and honeynets [75], while useful for capturing and analyzing malicious binaries, incur much too long of a time delay (often involving human intervention) to be practical for botnet mitigation. To address these limitations, several botnet detection approaches have recently been proposed, trying to discover botnets based on the network or host behavior typical of most bots. Since the IRC protocol [47] is currently one of the most popular C&C protocols used by botnets, most approaches to date focus on using some aspects of the IRC protocol for botnet identification, such as traffic monitoring and identifying IRC C&C servers. For example, Rishi [32] passively monitors IRC traffic for suspicious IRC nicknames, IRC servers, and uncommon server ports to detect bot-infected machines. Binkley and Singh [22] proposed detection of IRC-based botnets via TCP anomaly detection and IRC message statistics. BotHunter [35] attempts detection using IDS-driven dialog correlation based on IRC C&C communication and other common actions taken during the life cycle of a bot. Meanwhile, to track and analyze botnets in a large tier-1 ISP, Karasaridis *et al.* [49] proposed a wide-scale detection technique that looks for typical network-flow patterns between bots and their controllers. BotSniffer [36] identifies HTTP- and IRC-based C&C channels by capturing the coordinated and synchronized communication patterns in the C&C traffic. Unfortunately, because of their reliance on IRC- or HTTP-based C&C protocols, these detection schemes can potentially be subverted using encrypted channels or customized C&C protocols (e.g., P2P, FTP, etc.). To eliminate the reliance on

IRC- or HTTP-based C&C protocols for identifying botnets, Gu *et al.* proposed BotMiner [34], which clusters similar communication and malicious traffic and performs cross-cluster correlation to identify potential bot-infected hosts.

The consistently evolving nature of botnets has attracted significant interest from the research community, yielding several proposals to predict the design of future botnets that enables greater robustness, more efficient communication and better resilience against node failures and C&C channel shutdown. For example, Zou and Cunningham [76] proposed a two-stage reconnaissance technique for constructing and maintaining a honeypot-aware botnet that is capable of removing infected honeypots from its network. Hund *et al.* [45] focused their botnet design on strong encryption and protection mechanisms to guard against potential disruption techniques. In addition, researchers have also studied strategies of using other communication channels, such as email [65], to disseminate botnet commands, exploiting the lack of sufficient defense mechanisms in these protocols.

1.4.6 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter II describes our automatic, real-time detection system for detecting botnet-based hosting domains utilizing redirection on enterprise networks. Chapter III covers the analysis of how botnet-based hosting services advertise with DNS on a global scale. It compares this to other domain types, identifying potential differentiating features. Chapter IV utilizes detailed DNS data and formal models to determine if current botnet resources are capable of thwarting existing detection mechanisms by mimicking benign domains. It introduces new detection strategies that are more resilient to such mimicry attacks. Chapter V examines the future threat that botnets can pose in the quickly growing mobile environment. It evaluates the feasibility of highly mobile botnets utilizing only open and unencrypted WiFi networks—for an unprecedented level of

stealth—to perform C&C, DDoS attacks and spam attacks, comparing various AP-selection algorithms and exploring possible defenses. Lastly, Chapter VI concludes the dissertation.

CHAPTER II

RB-Seeker: Auto-detection of Redirection Botnets

2.1 Introduction

In this chapter, we focus on detecting bots (or other compromised systems) used in redirection/proxy scams. We term these bots *Redirection Bots* (RBs) and call the botnets they compose *Redirection Botnets* (RBnets). Since botnets are the primary source of such redirection endeavors, detecting computers partaking in suspicious redirection can provide a critical means of detecting these botnets. Furthermore, a botnet's versatility allows it to provide multiple criminal services, and hence, by detecting and mitigating RBnets, we can help deter the other malicious activities they may perpetrate.

Botnets are essentially an abundant source of disposable redirection servers/proxies, which serve as the front-end to malicious content hosted elsewhere—on anything from a powerful central server to another bot. Used as a misdirection mechanism for evading detection, RBnets are used in tandem with other criminal scams, constituting only a portion of the overall operation. For example, spam/phishing campaigns often utilize a RBnet for misdirection. They begin by using some spamming mechanism (e.g., a hijacked mail server and/or a botnet) to send seemingly interesting phishing emails. Within the phishing emails are innocuously disguised embedded links pointing to a RBnet. Once victims click the embedded links, they connect to the bots which then

redirect them to—or serve as proxies for—the actual host of the nefarious content, as shown previously in Fig. 1.1. While this single layer of redirection is the simplest case, it is common for criminals to employ multiple layers of redirection between the victim and the malicious content host. Botnets are an attractive redirection mechanism because if one is discovered and blocked, there is an ample supply of other bots to take its place, and the blocked bot can still be used for other villainous activities. The use of RBnets for spam/phishing campaigns is so successful at protecting the malicious-content hosts that criminals are beginning to centralize their operations. Numerous bots act as forwarding servers for the same phishing/scam campaigns, redirecting users to the same final-destination servers (called *motherships*) which host the illegal content. This strategy grants criminals a high level of anonymity via redirection while enabling easy centralized management.

While the RBnets can be used to deliver malicious content to victims via either redirection or proxy, redirection offers several financial and performance advantages over proxy in terms of content availability, resource utilization, and ease of management. Because botnets are composed primarily of compromised home computers with unreliable connectivity, it is common for them to unexpectedly go offline. If a botmaster is using bots as proxies to deliver malicious content to victims, the bots must remain online during the entire session. When a proxy bot unexpectedly goes offline, the connection between the victim and the source of the malicious content is severed. Using bots for redirection, on the other hand, is more resilient to connection failure because individual bots within the RBnet need to maintain connectivity only long enough to redirect the victim. As a result, the use of redirection will be more effective in terms of content availability. Moreover, individual bots experience more resource strain when used as proxies. Each bot must maintain connections with victims while serving as a proxy to the actual content host. Since bots are often less powerful compromised home computers, this limits the number of victims that

can be serviced by an individual bot. A botmaster can achieve better utilization of the botnet's resources by using redirection, as it is considerably less taxing on the compromised home computers composing the botnet. This greatly improves the financial gain achievable with the botnet, making it possible for botmasters to rent out a single RBnet for multiple criminal redirection infrastructures. Finally, as a side effect of proxy bots' poor utilization of resources, botmasters must be more diligent in management. They must ensure that enough bots are online and that they are intelligently dispersed across multiple DNS servers, such that the number of victims connecting to an individual bot is within its ability to function as a proxy.

For the above reasons, we propose a system called *Redirection Botnet Seeker*, or *RB-Seeker*, that detects RBnets based on their intrinsic network behavior patterns. Our contributions are three-folded. First, the system make uses of comprehensive and abundant data sources, including approximately two month's worth of: NetFlow records dumped from a core router of the campus network, spam emails from online and local spam archives, and DNS logs from two major local DNS servers. These rich, real-world data sources complement each other and provide RB-seeker with extensive views from multiple vantage points, resulting in a better detection rate of RBnets. Second, we design and develop several effective algorithms to exploit unique features of RBnets, such as their connection patterns, flow characteristics, DNS record behavior and typical involvement in spam/phishing attacks. As a practical implementation for enterprise networks, RB-Seeker comprises multiple network-monitoring subsystems, collaborating to identify malicious redirection infrastructures. The first subsystem takes advantage of the fact that redirection is often used for phishing/advertising. It analyzes embedded HTTP links in spam emails acquired from various sources using traditional spam detection systems and looks for links that redirect victims to different domains. The second subsystem improves the redirection detection capability by analyzing passive network traces at a large (i.e., of 40,000 students and several

thousand faculty and staff) university core router and exploiting the statistical difference between the connection patterns of redirection and normal browsing. Together, these two subsystems compile a database of redirection domains, which is used by the third subsystem that actively polls and monitors the DNS-query results for suspicious behavior. Third, we developed a 2-tier detection system that can detect both typical/aggressive and stealthy RBnets (i.e., those mimicking valid DNS behavior, such as Content Distribution Networks (CDNs)), which are likely participating in multiple botnet activities. In addition, as a behavior-based approach, RB-Seeker doesn't rely on malware signatures and is, therefore, more immune to traditional evasion techniques that are often and successfully employed by botmasters (e.g., polymorphism or malware packers).

The remainder of the chapter is organized as follows. We review the related literature in Section 2.2. Section 2.3 presents an overview of the system design and architecture. Sections 2.4–2.6 describe the three closely interacting subsystems of RB-Seeker, which cooperatively achieve accurate identification of RBnets based on their unique network and DNS behavior. Section 2.8 evaluates the effectiveness of RB-Seeker, and the chapter concludes with Section 2.9.

2.2 Related Work

In this chapter, we propose a novel detection technique for discovering botnets involved in redirection infrastructures. Our approach differs from previous work in that, instead of identifying botnet C&C channels, which can be evaded through modified or customized C&C protocols, we focus on the intrinsic behavior of RBnets. The behavior can be collected in real time from a variety of network-level traces, such as NetFlow, spam emails and DNS logs, regardless of the C&C protocols used by botnets. Similar network-level traffic analysis has also been widely used in both research projects and commercial products to combat malicious attacks, such as DDoS

and spam. For instance, the Internet Motion Sensor (IMS) [21] is a global monitoring system that utilizes many distributed sensors to monitor traffic to Internet dark spaces, capturing the scanning traffic of worm propagation. The major purpose of IMS is to detect random-scanning worms and prevent scanning traffic from reaching victims. Hence, it is not efficient for detecting botnets since not all scanning traffic originates from botnets, and botnets often propagate through other channels, such as email attachments (e.g., Peacomm bot), social engineering, browser vulnerabilities, etc. Cisco's Ironport [26] and P-cube [27] products investigate network (in particular SMTP) traffic as it passes through the systems. They use blacklist- or content-based filtering to detect spam emails and stop them before reaching the mail server. Because of the growing involvement of RBnets in spam/phishing campaigns [18], our approach can be easily integrated into these systems for practical deployment. The success of Ironport and P-cube in detecting incoming spam emails at the connection level will enable our approach to proactively detect RBnets and protect unsuspecting customers.

Redirection techniques have often been used for redirection web spams, where attackers try to boost their rankings in search engine results by presenting false content to indexing crawlers, automatically redirecting users' browsers to different, and often nefarious, web pages. Wu and Davision [73] studied the distribution of different redirection techniques. Chellapilla and Maykov [25] researched the prevalence of JavaScript redirection on the web and gave a detailed taxonomy of different JavaScript-based redirection approaches. They concluded that this type of redirection is the most notorious and difficult to detect due to the versatility of JavaScript, which allows for a number of obfuscation and dynamic-script-injection techniques. Wang *et al.* [72] built a system called "Strider Honeymonkey" to visit each page with a web browser and analyze the redirection behavior of malicious web servers. Their results showed that most malicious websites use front-end servers to automatically redirect

browser traffic to a back-end exploit server that specializes in exploiting client computers. Along the same lines, Spamscatter [18] mines the URL links the spam emails and follows any redirection to reach the destination scam websites. They found that over 68% of scams adopt certain redirection techniques to protect the true destination servers. Because of the large-scale and disposable nature of botnets, bots serve as ideal platforms for hosting redirection services. This is exemplified by the recent emergence of *fast-flux service network* (FFSNs) [60], which achieve high online availability by rapidly changing the IP addresses associated with the fast-flux (FF) domains. In FFSNs, most of the nodes are bots whose purpose is to redirect the unsuspecting users to the destination website hosting the phishing/scam content and/or exploit code. Holz *et al.* [39] studied the characteristics of FFSNs and developed detection algorithms that first extract URL links in spam emails and then identify FFSNs based on the number of unique IP addresses returned in DNS queries and the number of unique Autonomous Service Networks (ASNs) to which those IP addresses belong. This is very efficient in capturing FFSNs whose behavior is drastically different from the normal cases. RB-Seeker differs from these in several ways. First, RB-Seeker detects RBnets by utilizing more comprehensive data sources, including NetFlow, DNS-query logs and URLs in spam emails. This approach effectively alleviates the shortcomings of the spam-only approach. For instance, the URLs embedded a spam email could be heavily obfuscated or included inside a PDF or image. Furthermore, inspecting the content of all the emails is not always possible given privacy concerns. Second, RB-Seeker monitors multiple features of suspicious domains' DNS behavior over an extended period of time, utilizing an effective 2-tier detection strategy; this enables RB-Seeker to accurately detect RBnets with both aggressive and slowly changing (stealthy) DNS characteristics.

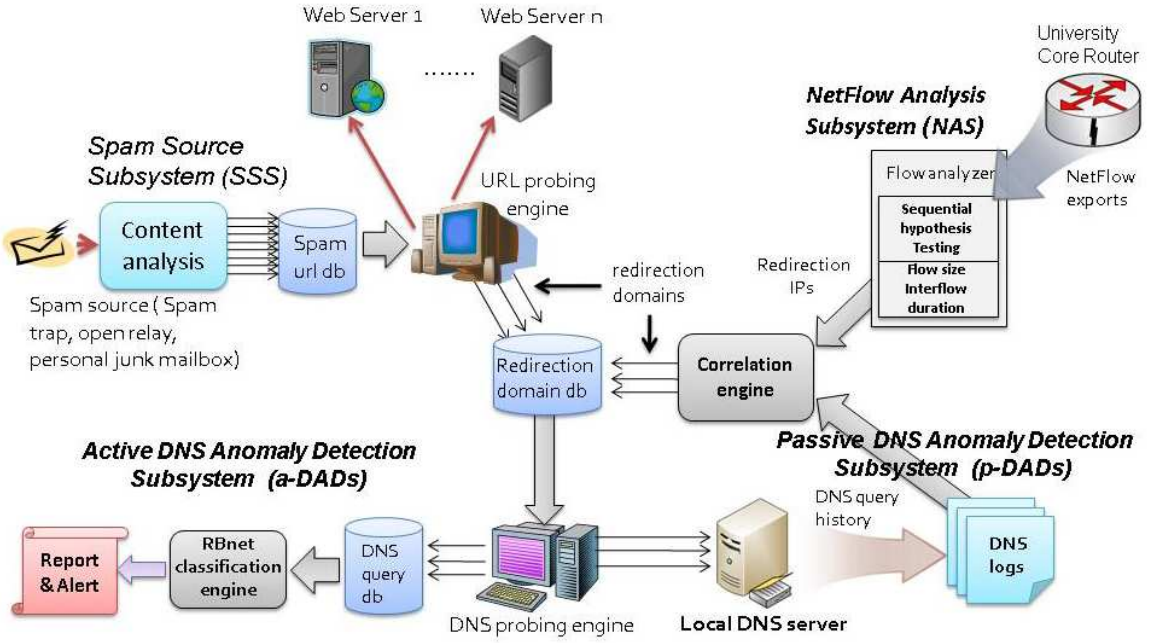


Figure 2.1: Architecture of the proof-of-concept RB-Seeker

2.3 System Architecture

We have developed and prototyped a system aimed at the automatic identification of suspicious redirection behavior and bot-infected computers involved in RBnets. Fig. 2.1 shows the architecture of the proof-of-concept RB-Seeker, which consists of three cooperative subsystems monitoring three primary input sources: spam emails (acquired from various sources using existing spam detectors), NetFlow data, and DNS server logs and query results.

The first subsystem, called the *Spam Source Subsystem* (SSS), consists of two components: a *content analysis* component and a *URL probing engine*. The former accrues a vast collection of spam emails using traditional spam detection techniques on personal spam mailboxes, online spam archives, and a spam relay server setup on a residential network. It analyzes the spam and extracts the embedded links into the *spam URL database*. The URL probing engine follows the embedded links stored in the spam URL database and compiles a list of domains participating in redirec-

tion, which are added to the *redirection domain database*. The second component, the *NetFlow Analysis Subsystem* (NAS), also generates a list of redirection domains. However, unlike the SSS, the NAS monitors network flows on a large university core router and uses sequential hypothesis testing to detect IPs participating in redirection based on flow characteristics. These IPs are fed into the *correlation engine*, which uses DNS-query logs to extract the associated domains, adding them to the redirection domain database. The redirection domain database compiled by the SSS and the NAS is used by our third—and final—subsystem, the *active DNS Anomaly Detection Subsystem* (a-DADS), which comprises two components. The first component, the *DNS probing engine*, continuously performs DNS digs on the domains in the redirection domain database, logging the results to the *DNS-query database*. The other component of the a-DADS, the *RBnet classification engine*, extracts various attributes for each domain from the DNS-query database and uses a hyperplane decision function to classify the domains as valid (i.e., benign) or malicious (i.e., belonging to a RBnet). If a domain is determined to be valid, it is removed from the redirection domain database and whitelisted to prevent the SSS or the NAS from reading it. When a RBnet domain is detected, it generates an alert report, containing the detailed DNS-query logs for the domain (stored in the DNS-query database) and allowing for further manual analysis on the RBnet's DNS behavior if needed.

2.4 Spam Source Subsystem

Because spammers are driven by an incentive for profits, most spam emails contain embedded phishing/scam links to lure unsuspecting individuals to phishing/scam websites or web pages containing malicious exploit code. In most cases, to protect the destination servers, users are redirected through one-or-more redirection servers, which are likely to be compromised computers serving as RBs. Taking advantage of this close connection between RBs and spam/phishing emails, the SSS uses the

embedded URLs in the spam message bodies as the starting point to trace out and detect machines participating in the redirection infrastructure. The SSS starts with the real-time collection of spam emails from multiple sources, including a spam relay trap set up at a residential network,[1] spam emails from the department mail server and several personal junk mailboxes at large email service providers. The SSS also downloads the latest spam emails from an online spam archive [37], which publishes more than 50,000 spam emails monthly. Upon collecting a new spam email, content analysis is performed on the message body to extract embedded URLs. After eliminating legitimate URLs using a precomposed whitelist, the SSS timestamps the remaining suspicious links and puts them into the spam URL database. The probing engine periodically retrieves URLs from the database and probes them to detect redirection behavior.

According to [25], modern browsers can be redirected in three ways: HTTP-status-code redirection (e.g., 301 moved permanently, 302 temporary redirect, etc.), HTTP-meta-refresh-header redirection, and client-side script redirection. The SSS handles HTTP-status-code redirection by using wget [5] to fetch the web page of a URL link. By default, wget detects the redirection status code and follows the URL specified in the location header entry of the HTTP response. By parsing the wget log file, the SSS can identify all intermediate redirection servers using HTTP-status-code redirection. Unfortunately, wget does not handle the other two types of redirection. To solve the problem, the SSS analyzes each downloaded web page and searches for HTTP refresh tags and redirection scripts. More specifically, to capture HTTP-meta-refresh-header redirection (where web pages use a META tag to redirect users), the SSS searches for the specific META tag with http-equiv attribute set to

[1] A spam relay trap works like a spam honeypot. It appears to be an open SMTP server that allows anyone on the Internet to send email through it (we block the outgoing traffic, so no spam emails are sent). Because spammers extensively scan and exploit open relays to re-route their spam emails, our open relay trap is able to collect a large amount of spam every day (on average 7,900 spam emails/day).

"refresh" and extracts the destination URL from the content attribute. For example, a typical use of META refresh header is as follows: `<meta http-equiv="refresh" content="0;url=http://www.destination.com">`. For script-based redirection, which is known to be most notorious and difficult to capture, the SSS focuses on detecting the most common JavaScript-based redirection techniques by scanning the web page for JavaScript code that changes a location property (e.g., "window.location," "location.replace," "document.location," "location.href," etc.) to the redirection destination URL. However, this type of static analysis can be evaded by sophisticated redirection techniques such as obfuscation, dynamic code injection and self-modifying JavaScript code [25]. We leave the dynamic analysis of web pages to identify redirection as our future work. However, a possible solution is to use a client-side honeypot (such as Capture-HPC [1]) that drives a real browser (with JavaScript enabled) to visit each suspicious URL and monitor the transition between web pages. After extracting the destination URL from either META headers or JavaScript code, the SSS invokes the probing engine again on these links to identify further redirection attempts. This procedure is repeated until no further redirection is identified in the final destination web page or a predefined threshold is reached (to prevent an infinite loop). The numbers of observed web pages that use each redirection technique are: 61,280 (54.1%) Status Code, 6,639 (5.9%) Refresh Tag and 45,285 (40.0%) JavaScript.

2.5 NetFlow Analysis Subsystem

This section describes the design of the second subsystem, the *NetFlow Analysis Subsystem* (NAS). Although quite useful, spam emails as a data source provide only a single vantage point with its own limitations. For example, spammers send image- or PDF-based spam emails to evade content-based filtering, so URL links might not appear as plain text in the spam body. Also, a user could be directed to a RB by clicking a link on a malicious web page, an IM message or many other ways.

In addition, inspecting each email body is not always possible because of privacy concerns/laws. To complement the SSS and improve the detectability of RBnets, the NAS takes advantage of NetFlow records, identifying redirection servers without the need for packet content analysis.

2.5.1 Redirection behavior characterization

Currently, the NAS operates on the NetFlow records collected from a core router of a very large university (the University of Michigan) network and looks for suspicious redirection attempts of web traffic. *NetFlow* is a network protocol developed by Cisco for summarizing IP traffic information [7]. Although capturing and analyzing packet-level data can provide the highest accuracy, the associated cost is prohibitively high even for a medium-size network. As a result, NetFlow, as a light-weight alternative, has become the most widely used technique for network monitoring, traffic accounting, etc. A *flow* is defined as a sequence of packets between a source and a destination within a single session or connection. A NetFlow record contains a variety of flow-level information such as IP protocol, source/destination IP and port, start and finish timestamps, and flow size. However, packet contents are not available, making it impossible to examine packet payloads and detect redirection behavior via HTTP status code or refresh headers. To address this limitation, we developed several redirection identification heuristics based only on the transport-layer information available in NetFlow data and the correlation of the traffic flow's size and timing behavior. The intuition behind these heuristics is that the behavior of visiting a redirection web server exhibits unique characteristics in terms of flow size, flow duration, and inter-flow duration, which are statistically different from normal, non-redirecting websites (see Table 2.1 for a detailed comparison) and can, therefore, be used to capture redirection activities. In Table 2.1, to obtain the "ground truth" of redirection behavior, we collected a set of server IPs that have been determined

		Mean	Median	Std dev
Flow duration (ms)	redirection	305.5	128.6	2159.2
	normal	33042.3	10028.8	91912.5
Inter-flow duration (ms)	redirection	392.7	154.4	872.4
	normal	40132.9	1345.5	87281.0
Flow size (bytes)	redirection	2401	629	44530
	normal	51495	4852	192431

Table 2.1: Comparison of average flow characteristics between redirection and normal browsing

by the SSS to perform redirection activities; we then use use tcpdump to capture all the packets between the SSS and those servers. In this way, we can build a database of redirection behaviors from the confirmed redirection-server IPs and compute the values for each feature. Similarly, the values for normal browsing are computed using two days' packet traces of a user's normal web browsing activities after removing the packets of identified redirection connections. Next, we will elaborate on each feature and the intuition behind it.

Short inter-flow duration Redirection often leads to multiple, consecutive HTTP flows from the same source IP address to different destination web servers within a short time period. The *inter-flow duration* is defined as the difference between the start times of two consecutive flows originating from the same source IP and destined for distinct destination IPs. Intuitively, the fast and automatic transition caused by redirection from one web server to another is in stark contrast to the considerably longer time taken for a user to move between websites during normal web browsing, e.g., by manually clicking the links. From Table 2.1, we can see that normal browsing usually takes two orders-of-magnitude longer than redirection.

Small flow size The flow size of visiting a redirection website is much smaller than that of visiting a normal website. This is because the redirection server usually returns only the redirection command data, such as HTTP status code, so that it will not waste bandwidth and, hence, can be used for redirecting more clients. For

example, in the case of most HTTP-status-code redirections, the redirection server returns only several tens of bytes, containing the status code (e.g., 301, 302) and a new destination server location. On the other hand, visiting a normal website usually necessitates downloading its homepage (often with pictures, longer texts, and embedded objects); thus, the flows of normal browsing are much larger in size (see Table 2.1).

Short flow duration Because of the small amount of data returned by a redirection server, the communication time (i.e., flow duration) between a user and a redirection server is often much smaller than for that of a valid web server. Since the purpose of a RB is to forward a client to the mothership hosting the nefarious content, there is no benefit for a RB to maintain the connection with the client longer than needed. In most cases, the RB terminates the connection as soon as the client is handed over to another web server. By contrast, the connection time for normal web browsing is considerably longer, especially because the current version of HTTP/1.1 introduces the keep-alive mechanism, which allows long-lived, persistent connections. For example, Internet Explorer (IE) times out a connection only after 60 seconds of inactivity.

2.5.2 Sequential hypothesis testing

Based on the above characteristics, the NAS exploits the temporal and size correlations among flows to identify redirection behavior. The NAS first sorts flow records chronologically and groups them by the source IP addresses. Within each group, the NAS computes the values of each feature—inter-flow duration, flow size and flow duration—for each destination IP; this forms an observed sample for each connection event between the source IP and a destination server. Our goal is then to classify whether the remote server is performing "redirection" or "normal" behavior. The simplest approach is to set up a fixed threshold for all three features and make decisions

based on each individual observation. However, as we will show later, all the features of normal and redirection behavior have heavy-tailed distributions, indicating that a simple threshold method may introduce significant classification errors. Intuitively, this could be improved by utilizing multiple observations so that each decision is made with a high level of confidence. To achieve this goal, we adopt Sequential Probability Ratio Testing (SPRT) [70]—a type of statistical hypothesis testing where the number of observations required by the test is not predetermined but is a random variable determined by the underlying distribution. In other words, a decision is made only after enough evidence has been accumulated to support the acceptance or rejection of the hypothesis. Thus, SPRT achieves high accuracy and has been widely used in many anomaly detection scenarios, such as detecting port scanners [48] and botnets [36].

We consider two hypotheses when performing Sequential Hypothesis Testing (SHT): H_0 (the remote server is a normal server) and H_1 (the remote server is a redirection server). In order to demonstrate how SHT works, let's examine how the NAS uses it to arrive at a classification decision for the inter-flow-duration feature (the procedure is identical for the other two features). Assuming the hypothesis H_i holds, the inter-flow duration follows some distribution (how to model such a distribution is discussed in the next subsection) whose density function is denoted as $f_i(T_{inter}) = f(T_{inter}|H_i)$. Let $T_1, T_2, \ldots, T_n$ be a sequence of observed samples of the inter-flow duration for the same destination IP. We can compute the likelihood ratio as $\Lambda(n) = \frac{f_1(T_2)f_1(T_2)\cdots f_1(T_n)}{f_0(T_2)f_0(T_2)\cdots f_0(T_n)} = \prod_{k=1}^{n} \frac{f_1(T_k)}{f_0(T_k)}$. Then, for each stage k, or the k-th observation ($k = 1, 2, \ldots, n$), the test leads to one of three decisions based on the following decision rules: (1) accept H_1 if the likelihood ratio exceeds the threshold η_1; (2) accept H_0 if the likelihood ratio is below another threshold, η_0; and (3) otherwise, pend and wait for another observation. More specifically, for the k-th observation of

a new connection:

$$\text{Output} = \begin{cases} \text{Accept } H_1 & \text{if } \Lambda(n) \geq \eta_1 \\ \text{Accept } H_0 & \text{if } \Lambda(n) \leq \eta_0 \\ \text{Pend} & \text{otherwise.} \end{cases}$$

One nice property of SHT is that the thresholds η_0 and η_1 can be set according to the target false-positive rate α (type-1 error: reject H_0 although it is true) and false-negative rate β (type-2 error: accept H_0 although it is false). Wald [70] showed that, by setting the threshold to $\eta_0 = \frac{1-\beta}{\alpha}$ and $\eta_1 = \frac{\beta}{1-\alpha}$, the true false-positive and false-negative rates will deviate from α and β by only a small margin.

2.5.3 Flow-based redirection identification

Fig. 2.2 shows the flowchart of how the NAS combines the three features and applies SHT to detect redirection servers. When a new connection is observed from a source IP (assuming the new connection is the n-th observation), the inter-flow duration T_n (defined as time difference between the current flow and the immediately preceding flow from the same IP) is compared against a loose threshold; this threshold value is chosen so that any inter-flow duration larger than this threshold is very unlikely to have been caused by redirection.[2] Then, if the inter-flow duration is below the threshold, the hypothesis testing history $\Lambda(n-1)$ is retrieved from the database, and the new likelihood ratio is computed as $\Lambda(n) = \Lambda(n-1) * \frac{f_1(T_n)}{f_0(T_n)}$. Depending on the likelihood ratio, the NAS outputs one of three decisions: accept H_0, reject H_0 (i.e., accept H_1), or pend. If the output is to accept H_0, then the destination IP of the preceding flow is considered a normal server, and the hypothesis testing history for that IP is cleaned up. If the existing data samples cannot provide enough confidence to reject or accept the hypothesis, the pending decision is chosen, and the current likelihood ratio is stored in the *hypothesis testing database* for future testing when additional observations become available. Finally, if the output suggests that

[2] In our current experiment, we set this threshold to 30 seconds.

a potential redirection behavior has been observed (i.e., to accept H_1) according to the inter-flow duration, a second hypothesis testing is performed on the flow size of the preceding flow. The reason why the NAS relies on multiple metrics to identify redirection behavior is that a single metric often leads to false positives. Specifically, the inter-flow-based hypothesis testing cannot distinguish concurrent flows from redirection flows. Concurrent flows occur when the destination web server references resources (e.g., pictures, videos) from other servers. As a result, when the client browses the web page, it requests several concurrent connections to multiple destinations within a short time frame. This results in short inter-flow durations that are indistinguishable from those caused by redirection behavior. Thus, we use flow size as a second-line filter to eliminate the potential false positives resulting from concurrent flows. Because the purpose of concurrent flows is often to fetch the (multimedia) contents of a web page, the flow size is expected to be much larger than is needed for redirection commands (e.g., status code). The hypothesis testing on flow size determines whether to accept the hypothesis or store the likelihood ratio for future testing. If the result indicates redirection behavior, then a third, optional, hypothesis testing on flow duration could be performed. The flow duration is optional because our experimental measurements have shown that some redirection servers do not terminate connections—even after sending the redirection status code. The idle connection is kept alive without any data transmission until the client browser times out and closes the session. We conjecture this could be due to misconfiguration of the server. Thus, if a more strict detection algorithm is desirable, the optional flow-duration hypothesis testing can be performed to reduce false positives at the cost of increasing the false-negative rate (e.g., the NAS may fail to detect long-lived redirection servers). In our current implementation of the NAS, only the first two hypothesis tests are performed.

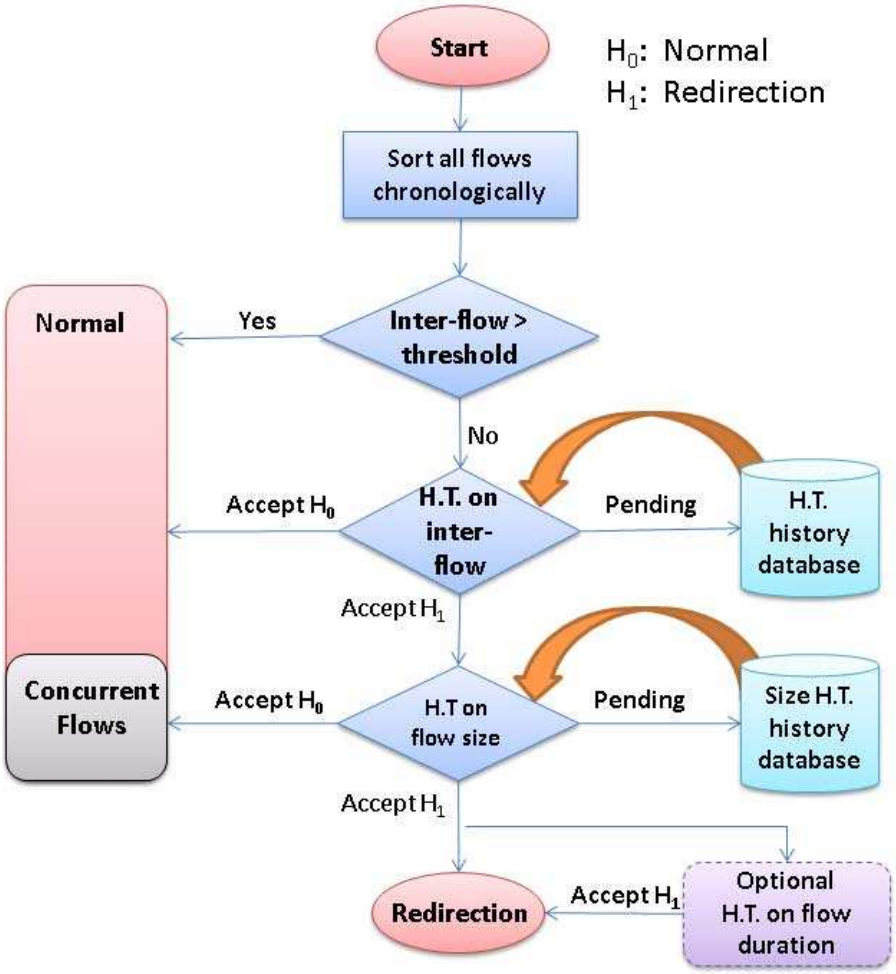


Figure 2.2: Flowchart of the algorithm for identification of redirection behaviors

2.5.4 Modeling the distribution of flow features

One of the prerequisites for a hypothesis test is to determine the density function of different features conditioned on the hypothesis, i.e., $f_i(T)$ and $f_i(S)$, where T is the inter-flow duration, S is the flow size and $i = 0$ or 1. As mentioned before, to obtain the "ground truth" of redirection behaviors, we collect packet traces of confirmed redirection servers from the SSS and normal web-browsing activities to build two (i.e., normal and redirection) datasets for each feature. A simple examination of the histogram of these datasets shows that all the features follow non-negative, heavy-tailed distributions, each with a single tail. Statistical distributions that satisfy these conditions include Pareto, log-normal and Weibull distributions. We apply

	μ	95% C.I. of μ	σ	95% C.I. of σ
Inter-R	5.270	[5.260, 5.281]	0.974	[0.966, 0. 9812]
Inter-N	7.982	[7.896, 8.067]	2.512	[2.454, 2.574]
Size-R	6.529	[6.517, 6.542]	0.956	[0.948, 0.965]
Size-N	8.423	[8.380, 8.466]	2.093	[2.063, 2.125]

Table 2.2: Maximum likelihood estimates of parameters for a log-normal distribution (Inter-R means inter-flow duration for redirection, and Inter-N means inter-flow duration for normal browsing; Size-R and Size-N are defined similarly.)

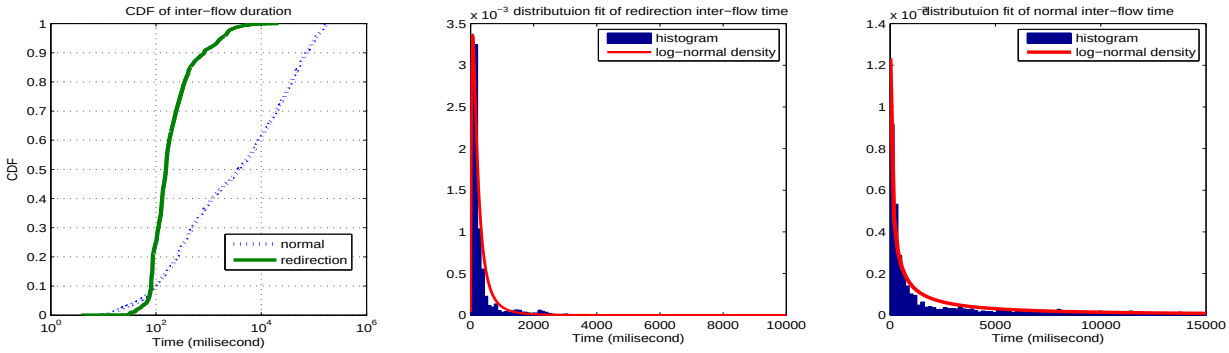


Figure 2.3: Log-normal distribution fit for inter-flow durations

the maximum likelihood (ML) method to estimate parameters for each distribution and compute Kolmogorov-Smirnov statistics [6]—a popular method to evaluate how well a distribution fits the actual data. The results reveal that the log-normal distribution achieves the best fit between the empirical data and analytical model. Its density function is given in the form of: $f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(lnx-\mu)^2}{2\sigma^2}}$. The log-normal distribution is characterized by two parameters, μ and σ. Table 2.2 shows the ML estimates of these two parameters and their 95% confidence interval for inter-flow and flow-size features. Fig. 2.3 depicts the CDF of inter-flow durations in both redirection and normal cases as well as the log-normal distribution fitting the results of the ML estimation. The flow-size result is similar to this and, hence, omitted. Having estimated μ and σ, the hypothesis tests on these features can be done easily by calculating the likelihood ratio with the density function of a log-normal distribution and the parameter values in Table 2.2.

2.5.5 DNS log correlation

Using continuous monitoring of traffic flows, the NAS performs SHT to detect potential redirection activities and stores the IP addresses of suspected redirection servers (Fig. 2.1). However, many redirection servers could be benign since redirection is also frequently used for legitimate purposes (e.g., website migration, the use of a short and easily-remembered domain name to replace a long and convoluted one, redirection among alias domain names, etc.). To pinpoint malicious RBnets, we need to validate the DNS behavior of their domain names. However, NetFlow records only store the flow IP addresses without their domain names. Note that the reverse DNS lookup is not useful in identifying the domain names for RBs. The forward mapping between the phishing/scam domains and bots' IPs are registered by the adversaries and are resolved by DNS servers they possibly control, allowing attackers to associate an arbitrary domain name with their bots' IPs. On the other hand, a reverse DNS lookup returns the actual domain name of the RB as determined by the bot's Internet Service Provider (ISP); thus, it will not match the malicious domain used in the scam. To address this problem, the NAS correlates the redirection IPs it has detected with domains found in the local DNS servers' DNS-query logs. These identified redirection domains will first be filtered against a known whitelist to remove legitimate redirection domains, such as popular CDNs[3] (e.g., Akamai, CoDeeN, LimeLight, etc.) and known redirection service domains [4] (e.g., google, yahoo, tinyurl, etc.). The remaining domains are placed into the redirection domain database to be probed and verified by the a-DADS, which we discuss next.

[3]We also developed an effective heuristic to detect previously unseen CDN domains and IPs, which will be discussed in Section 2.6.3.

2.6 Active DNS Anomaly Detection Subsystem

The SSS and the NAS identify domains involved in redirection, either deterministically (from spam emails) or probabilistically (from NetFlow records), and store them in the redirection domain database. However, since valid domains commonly make use of redirection (e.g., to balance server load), there is no guarantee that the redirection domains detected belong to a RBnet. It is the purpose of the *active DNS Detection Subsystem* (a-DADS) to determine if any of the suspicious domains in the database actually belong to a RBnet.

2.6.1 Data collection and analysis

For each unique domain in the redirection domain database, the a-DADS continuously performs and logs DNS queries for the domain's IPs (A records), name servers (NS records), name servers' IPs (NS-A records), the reverse DNS lookup on any IPs returned (i.e., the A and NS-A records), and the Autonomous System Number (ASN) to which each IP belongs. To analyze RBnet behavior over time, we continue to perform these digs until we have obtained at least a week's worth of *valid queries* (i.e., non-cached queries that didn't time out).

2.6.2 Characterization of RBnet behavior

RBnets, by their very nature, exhibit atypical DNS behavior. This is due to the way a RBnet is structured and the function it serves. A criminal, utilizing a redirection infrastructure for misdirection, will register an arbitrary domain name—perhaps a misspelling of a popular domain or an innocuous name used for a phishing email—and then point it to several bots in the RBnet. Thus, when the victim tries to visit the malicious domain, the DNS server will respond with some of the many bots' IPs, redirecting the victim. In order for this mechanism to provide reliable content delivery for the malicious domain, the botmaster must make certain the bots

registered with DNS for the malicious domain are online. Otherwise, the victim will not be able to connect to the bot and be redirected to the nefarious content. Botnets naturally suffer from unreliable connectivity, since they are typically comprised of less secure home computers which are not always online. Even with increased use of 'always-online' broadband Internet, home desktops and laptops are often turned off or suspended, making them unreliable. To overcome this shortcoming, the botmaster must take certain measures to ensure the domain resolves to one of the online RBs, resulting in abnormal DNS-query behavior. Based on the mechanisms available to the botmaster through DNS, we expect to observe behavioral abnormalities for the following attributes.

IP usage Botmasters incorporate several IP-management strategies when advertising their RBnets to the DNS. These strategies cause the DNS-query results for RBnet domains to exhibit discernible variations from those of valid domains using CDNs or Round-Robin DNS (RRDNS). First, we expect there to be more unique IPs associated with a particular RBnet domain over time. RBnets will accrue, over time, more unique IPs than valid domains, since valid domains will have more stable servers hosting the content. In addition to supplying more IPs than valid domains over time, we expect many RBnets to supply more unique IPs per individual *valid query*. By supplying a larger set of IPs per query, the botmaster helps ensure the malicious domain resolves to a valid IP. With a larger pool of IPs, there is a higher probability one of them belongs to an online bot, decreasing the level of vigilance required in monitoring the RBnet's connectivity. As a further consequence of poor connectivity, botmasters will have to replace the IPs registered to their malicious domain frequently, requiring short TTL values. While CDNs also replace their IPs frequently, they will have a smaller pool of unique IPs over time than RBnets due to their use of more stable servers.

Reverse DNS lookup This involves a reverse DNS query (i.e., PTR records) on the IPs returned in the A records. While a reverse DNS lookup doesn't always return a result, when it does, it can be used to help detect a RBnet. Specifically, RBnets will often return domain names with "bad words" typical of home computers, such as cable, broadband, comcast, charter, dialup, dynamic, etc. Therefore, for each domain, we rank the occurrences of suspicious words. This is a reliable metric since the domain names returned by a reverse DNS lookup (i.e., reverse DNS names) are set by the the bots' ISPs and cannot be easily faked by botmasters. For this reason, reverse DNS queries on compromised home computers will often return domain names littered with suspicious words not present for valid domains (both RRDNS and CDNs). We also filter out valid domains containing "bad words" (e.g., comcast.net, charter.net) so that these are not unfairly weighted.

AS count Because the compromised computers composing botnets are scattered geographically, the IPs returned for RBnet domains will belong to a more diverse set of Autonomous Systems (ASes). Thus, we keep track of the number of unique ASes associated with a domain, as it should be a helpful metric in identifying RBnets.

2.6.3 CDN Filter

Consisting of thousands of servers distributed around the globe, CDNs must assume that any (and potentially many) of their servers could experience downtime due to network, software, or hardware failures. With this pragmatic view in mind, CDNs have been developed to be resilient to such failures, ensuring reliable content availability to their customers [16]. Consequently, they utilize DNS-based solutions similar to those currently being employed by botmasters. For example, CDNs and RBnets both use very small TTL values, allowing their networks to quickly respond to failure. Also, they both often advertise multiple IP addresses for a given domain, hedging their bets should some IP addresses go offline. CDNs also make use of ag-

gressive load balancing, frequently changing the IPs advertised by DNS to ensure the highest throughput for their customers. These techniques often make the DNS behavior of CDNs appear akin to that of a RBnet.

Because of this behavioral similarity among CDNs and RBnets, we have developed a *CDN Filter* for the a-DADS to remove—from the redirection domain database—those domains that we can determine to be using legitimate CDNs. The CDN Filter operates based on the following two observations: (1) RBnet domains do not return IPs for legitimate CDNs in their DNS A records, and (2) each CDN server (with its corresponding IPs) will be used to service multiple legitimate domains. As a result, the a-DADS's CDN Filter analyzes the reverse DNS lookup of the A-record IPs for all the domains in the redirection domain database. When an A-record IP displays a reverse DNS name matching that of a legitimate CDN, we add the IPs seen for that domain to the *CDN-IP database*. The CDN-IP database is then cross-referenced against the IPs seen for other domains in the redirection domain database. When a domain is discovered to contain an IP from the CDN-IP database, it is flagged as using a CDN, and its IPs are added to the CDN-IP database. This process repeats, filtering out those domains that are using valid CDNs. In this way, the a-DADS's CDN Filter removes those valid (non-RBnet) domains from the redirection domain database that exhibit DNS patterns most similar to those of RBnets.

2.6.4 RBnet classification

After filtering out the known CDNs with the CDN Filter, the a-DADS employs a *2-tier*, linear Support Vector Machine (SVM) detection strategy on the remaining suspicious domains. The first-tier SVM (SVM-1) is designed to quickly identify those RBnets exhibiting a strong deviation from normal DNS behavior, which we term *Aggressive RBnets*. Any domain not identified as a RBnet by SVM-1 is further analyzed. The a-DADS continues to perform digs on the domain and applies

the second-tier SVM (SVM-2) to the results. While SVM-1 is designed to identify Aggressive RBnets quickly from minimal valid DNS queries, SVM-2 takes more time and is capable of detecting RBnets that try to mimic the short-term DNS behavior of valid domains, which we term *Stealth RBnets*.

Both SVM-1 and SVM-2 make use of a linear classifier of the form:

$$F(x) = \begin{cases} w^T x - b > 0 \text{ , if } x \text{ is valid domain} \\ w^T x - b < 0 \text{ , if } x \text{ is RB domain} \end{cases}$$

where w is a weight vector, b is a bias term, and x is a vector of behavioral attributes. These variables and vectors are different for each tier, and will be discussed next.

2.6.4.1 SVM-1

Using the CDN Filter, we filtered out any known CDN domains from the redirection domain database compiled by the SSS and the NAS. After filtering, the remaining suspicious domains were predominantly RRDNS, with a few CDN domains that escaped detection by our filter. We carefully selected a set of 124 valid domains that were representative of the different types of valid DNS behavior we observed. We also manually identified 18 Aggressive RBnet domains, which were easy to identify by hand due to their aggressive IP management tactics. These 142 domains (124 valid domains and 18 Aggressive RBnet domains) composed the *training set* for SVM-1. We then used 10-fold cross-validation on the training set to determine which behavioral characteristics best differentiated RBnets from valid domains based on only two valid queries. We discovered that three behavioral characteristics dominated the SVM equation: the total number of unique IPs seen, the total number of unique ASes seen, and the number of reverse DNS names with "bad words". The other behavioral characteristics we previously mentioned, while useful when analyzing multiple queries, were not as significant when observing only two valid queries. We chose to

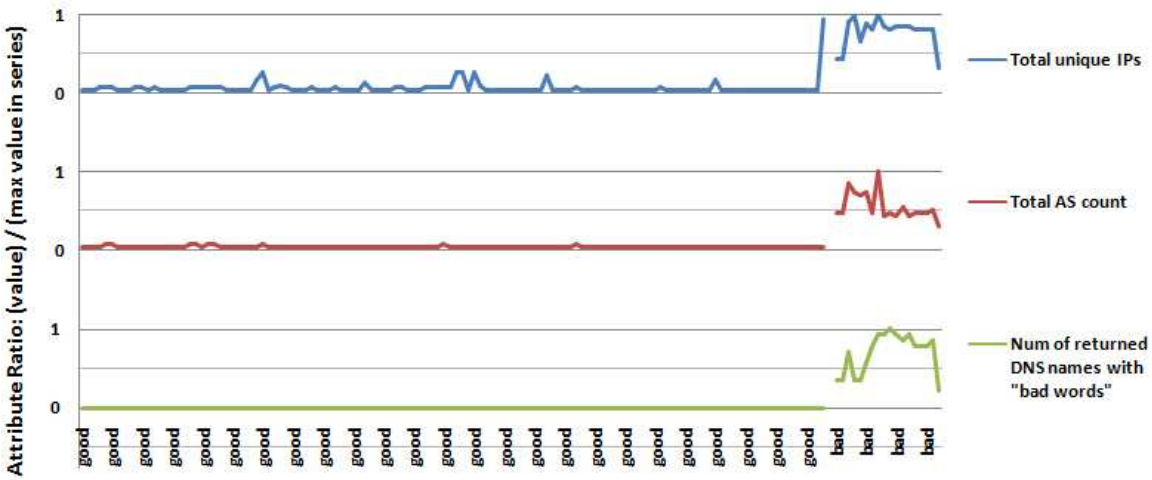


Figure 2.4: Domain attributes for the 142 domains in SVM-1 training set (two valid queries)

use the linear SVM best suited for classification based on the minimal number of valid queries, since the goal of SVM-1 is fast, accurate detection of Aggressive RBnets. The resulting equation is used for SVM-1, with the value returned being indicative of a domain's suspicion level:

$$\begin{aligned} f(x) &= w^T x - b \\ &= -1.257 * N_{unique\_IPs} - 26.401 * N_{ASes} \\ &\quad -13.024 * N_{DNS\_bad\_words} + 162.851 \end{aligned}$$

where N_{unique_IPs} is the number of unique IPs, N_{ASes} is the number of ASes, and $N_{DNS\_bad\_words}$ is the number of DNS "bad words" seen (should the reverse DNS lookup return a result). Testing the SVM-1 equation by using 10-fold cross-validation on the training set achieved 99.3% accuracy. We further evaluated the accuracy of SVM-1 by running it on the remaining domains in the redirection domain database *not* used in the training set; the evaluation of these results will be discussed Section 2.8.2.

A graph of the three attributes used in SVM-1 can be seen in Fig. 2.4. Each attribute is represented as a fraction of the largest value seen for that attribute among

all the domains in the training set, allowing us to show all their relationships on a scale from 0 to 1. From Fig. 2.4, it is clear that Aggressive RBnets display a distinct behavioral difference from valid domains for the monitored attributes (gaps in the graph visually separate valid domains from Aggressive RBnet domains). The spike at the end of the good domains for the *Total unique IPs* is due to a CDN that managed to escape our CDN Filter. While it contains a large number of IPs (on par with Aggressive RBnets), there is a noticeable difference in its number of ASes and DNS "bad words". This difference allows SVM-1 to classify it as benign, causing it to be further monitored by SVM-2. Should any RBnets exhibit behavior similar to valid CDNs, they also will be monitored by SVM-2, which takes advantage of long-term DNS behavior to distinguish valid domains from RBnets.

2.6.4.2 SVM-2

One factor that will differentiate a Stealth RBnet from a valid domain is the number of unique IPs and ASes it accrues *over time*. While DNS queries for valid domains (such as some CDNs) may return many IPs spanning multiple ASes (similar to RBnets), queries will continue to return those same IPs after a significant period of time. That is to say, the number of unique IPs and ASes returned by a valid domain over a day will be nearly the same set of IPs and ASes returned a few days later. This is because valid domains have fairly stable servers. While hardware or software failures may result in a server temporarily going offline, causing a new IP to be introduced to the DNS, they will not remain offline indefinitely. Ultimately, the problem will be fixed, the server brought back online, and its IP reintroduced into the DNS. On the contrary, Stealth RBnets are only able to *appear* like valid domains; they are still composed of compromised computers. The compromised computers may be more persistent than those in Aggressive RBnets, but they will still be more unreliable than the servers used in valid domains, such as CDNs. Additionally, some

Stealth RBnets may utilize a legitimate redirection infrastructure that has been compromised, allowing them to mimic valid domains more easily. However, the servers in the compromised redirection infrastructure will still be less persistent than a valid CDN for the following reasons. First, the system administrators of the legitimate redirection infrastructure might thwart the botmasters' abuse of their system, rendering some of the botmasters' IPs useless. Second, in an effort to remain undetected by the system administrators, the botmaster will have to continuously change which servers are being exploited for the Stealth RBnet. In either case, the Stealth RBnet will slowly, over time, continue to accrue more and more unique IPs that span more and more ASes. This is in direct opposition to a valid domain or CDN, which has a fairly stable pool of server IPs to advertise to the DNS.

From our manual analysis of Stealth RBnets, we discovered that they tend not to return reverse DNS names. This could be because they are not composed of home computers (which tend to return reverse DNS names more often than legitimate servers) or are utilizing legitimate redirection infrastructures that have been compromised. Additionally, they tend to show very little variance in unique IPs and ASes between valid queries. We discovered this is because they are utilizing very short TTL values of around one second. This allows the botmaster to use a single IP (or a small set of IPs) for multiple valid queries. The incredibly small TTL provides the botmaster with a fine level of control, permitting the IP to be changed as soon as the bot goes offline. In this way, the botmaster can keep both the unique IP count and the number of ASes low across multiple, valid queries, allowing the Stealth RBnet to go undetected by our SVM-1 as well as traditional FFSN detectors. To counter this strategy, our SVM-2 monitors the number of unique IPs and ASes seen in a day. It then continues to monitor the suspicious domain for up to a week, analyzing how many unique IPs and ASes it has accrued after this time span.

For the SVM-2's training set, we removed the 18 Aggressive RBnet domains from

the SVM-1's training set and replaced them with 10 Stealth RBnet domains, which we identified manually. We then used 10-fold cross-validation on the 134-domain training set (124 valid domains and 10 Stealth RBnet domains) to determine the behavioral attributes best suited for differentiating Stealth RBnets from valid domains, given an extended observational period. As expected, the previously mentioned attributes based on changes between valid queries became insignificant due to the very short TTL values imposed by botmasters. Additionally, the reverse DNS names with "bad words" became insignificant because none of the Stealth RBnets returned reverse DNS names. Thus, we found that SVM-2 only needed to monitor the number of unique IPs and ASes seen over time, for up to 1 week. We tested these metrics using 10-fold cross-validation on the training set and achieved 96.7% accuracy. We further evaluated the accuracy of SVM-2 by running it on the remaining domains in the redirection domain database *not* used in the training set; the evaluation of these results will be discussed in Section 2.8.2. The resulting linear equation is used for SVM-2, with the result indicating a domain's suspicion level:

$$\begin{aligned} f(x) &= w^T x - b \\ &= 52.497 * N_{DAY\_unique\_IPs} - 63.109 * N_{WEEK\_unique\_IPs} \\ &\quad -10.924 * (N_{DAY\_ASes} + N_{WEEK\_ASes}) + 227.985 \end{aligned}$$

where $N_{DAY\_unique\_IPs}$ is the number of unique IPs seen after a day, $N_{WEEK\_unique\_IPs}$ is the number of unique IPs seen after a week, N_{DAY_ASes} is the number of ASes seen after a day, and N_{WEEK_ASes} is the number of ASes seen after a week. Fig. 2.5 shows a graph of these four attributes for a subset of SVM-2's training set. While some good domains slightly increase their total unique IP count from a day to a week, the increase is not nearly as drastic as with Stealth RBnets. Furthermore, all of the good domains have a constant number of ASes over the week, whereas most of the Stealth RBnets display a slight increase. Also, from Fig. 2.5, it is apparent that during the first day, the Stealth RBnets and the good domains share similar behavioral attributes. It is

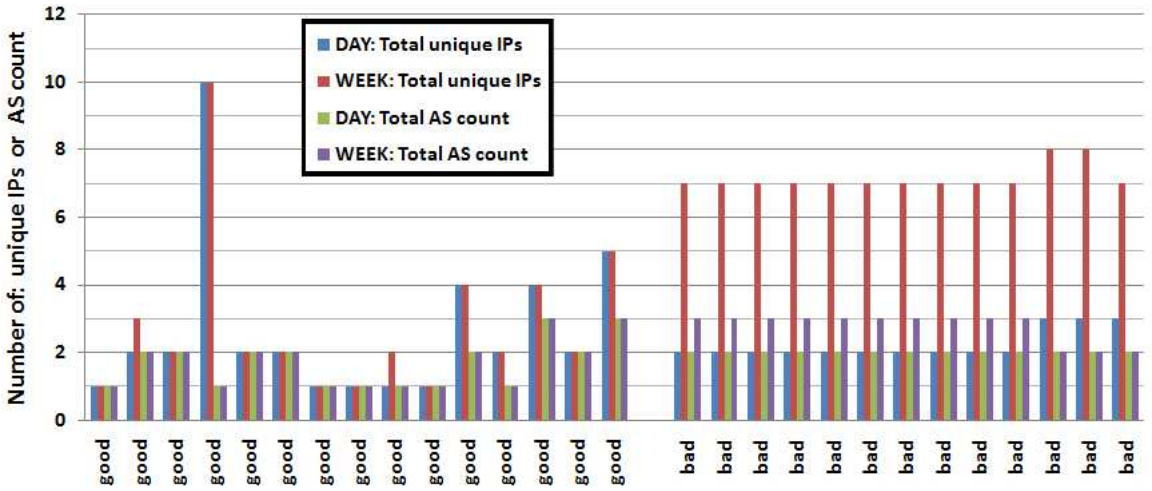


Figure 2.5: Domain attributes for subset of good and bad domains in SVM-2 training set

only after monitoring for an extended period of time that the Stealth RBnets show their true colors, demonstrating the need for SVM-2's extended monitoring window.

2.7 Discussion

Thus far, we have described the architecture of RB-Seeker and its effectiveness in detecting current RBnets. However, security solutions are in a constant arms race between defenders and attackers, and RB-Seeker is no exception. In this section, we discuss several ways adversaries may attempt to evade RB-Seeker, providing potential countermeasures against them.

An attacker or botmaster who has learned RB-Seeker's detection schemes may try to evade or confuse them by altering the RBnet's behavior according to the features used by the NAS, the SSS and the a-DADS. For instance, adversaries may try to confuse the NAS by invalidating the basic assumption that the NAS has made for redirection activities. Specifically, RBs may attempt to mimic normal, non-redirection servers by waiting for an extended period of time (e.g,. 30 seconds) before redirecting clients, creating a longer inter-flow duration. They may also try sending useless

content in their packets in addition to redirection commands, increasing the flow size. This may force the NAS to delay the detection decisions in order to accumulate enough observational samples. In the worst case, the NAS may misclassify the redirection activities as normal. Like most behavior-based detection systems, the NAS is vulnerable to mimicry attacks, where adversaries successfully disguise their behavior as normal activities. However, because the characteristics of redirection are generally two orders-of-magnitude smaller than those of normal browsing (Table 2.1), in order to mimic the normal behavior, the attacker has to use most of the bot's already limited resources. For example, bots must keep connections alive and send useless data, which will limit the number of victims that can be served by each individual bot. Otherwise, their consistent deviation from normal activities will still present a high probability of being caught by the NAS after observing enough samples. Second, to prevent the SSS from automatically extracting HTTP links from the email body, attackers may embed obfuscated/encoded URL links in spam emails instead of using plaintext or HTML format. They could also take advantage of sophisticated redirection techniques (e.g., obfuscated JavaScript) to circumvent the redirection detection engine in the SSS. Although our prototype implementation only handles the most common and simple URL formats and redirection techniques, the SSS can be easily strengthened to counter such evasion tactics by incorporating existing methods for analyzing text embedded in images [31] and detecting sophisticated redirection links with client-side honeypots [1]. Finally, to circumvent the a-DADS detection, a botmaster may attempt to mirror the DNS behavior of popular CDNs by reducing the number and diversity of IPs associated with the domain. However, as discussed earlier, this not only limits the availability and throughput achievable by the RBnets, but these Stealth RBnets can still be detected with the a-DADS's improved CDN filtering technique and 2-tier detection strategy. Therefore, while there are several ways a botmaster could attempt to evade detection, some of them are too expensive

to provide enough incentive to botmasters. Furthermore, as is demonstrated in the next section, RB-seeker is still quite effective in identifying many RBnets.

2.8 Implementation and Evaluation

In this section we describe the implementation of the overall system and evaluate the overhead of its subcomponents. We then evaluate the performance of the a-DADS classification function, comparing it with the current state-of-the-art. Lastly, we describe some of the DNS behavior for the RBnets detected with RB-Seeker.

2.8.1 Implementation and overhead evaluation

We implemented a proof-of-concept RB-Seeker for Linux kernel 2.6.18 on an HP ServerBlade with 2 Dual-Core AMD Opteron(tm) Processors (2.2 GHz, 2024 KB cache), 4 GB of RAM, and 260 GB of disk space. The subcomponents were implemented in Perl and Python. They were continuously run to extract redirection domains from spam emails and NetFlow traces and perform DNS queries on the suspect domains.

On average, the SSS analyzes approximately 10,000 spam emails daily (80% from spam relay and 20% from the spam archive and personal junk mail boxes) and extracts 9,000 unique URL links. Among them, the SSS applies the techniques described in Section 2.4 and determines more than 700 redirection domains daily, adding them to the redirection domain database. Meanwhile, the NAS receives 95,000,000 flows from the core router daily, 6,974,015 of which are HTTP flows[4] and are analyzed by the SHT algorithm (described in Section 2.5) to identify redirection activities. On average, the NAS identifies between 500 and 600 domains daily. We also test the processing speed of the NAS; the results show that the NAS is capable of parsing one day's HTTP flow data within 10 minutes, demonstrating its efficiency and suitability

[4]We consider a flow an HTTP flow if its destination port is 80 or 8080. Since no packet payload information is available, we are not able to detect HTTP flows using non-standard ports.

for online analysis. Another important factor influencing the NAS's speed in detecting a redirection server is the number of flows (i.e., observational samples) needed for the SHT algorithm to make a decision (i.e., accept or reject a hypothesis). Since the required number of observed samples in sequential testing is a random variable, depending on both current and historical observations [70], the expected number of observations (flows) for the NAS to determine if the destination IP is performing redirection can be approximated by:

$$E[N|H_1] = \frac{\beta \ln\frac{\beta}{1-\alpha} + (1-\beta)\ln\frac{1-\beta}{\alpha}}{E[\ln\frac{f_1(x)}{f_0(x)}]} = \frac{\beta \ln\frac{\beta}{1-\alpha} + (1-\beta)\ln\frac{1-\beta}{\alpha}}{\ln\frac{\sigma_0}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_0)^2}{2\sigma_0^2} - \frac{1}{2}}$$

where μ_i and σ_i $(i = 0, 1)$ are parameters for a log-normal distribution on the condition of normal browsing (H_0) and redirection (H_1). The values of μ_i and σ_i can be found in Table 2.2. As a result, the expected number of observations depends only on the target false-positive rate (α) or false-negative rate (β). Intuitively, if we want to reduce α and β, the expected number of required flows will increase because the NAS has to accumulate more observed samples to reach the desired confidence level before making a decision. Figs. 2.6 and 2.7 depict $E[N|H_1]$ with different values of α and β based on inter-flow duration and flow size, i.e., the expected number of flows the NAS has to observe in order to accept the hypothesis that the destination server performs redirection. One can see from the figures that the NAS is able to detect redirection servers by using only a small number of observed samples (normally 5 or 6) with low false-positive and false-negative rates.

In addition, because the a-DADS can perform queries on multiple redirection domains simultaneously, we split its functionality into two parts to keep its overhead and memory footprint small. The first part simply reads the most recent domains in the redirection domain database (built by the SSS and the NAS) and performs queries on the domains, logging the results to the DNS-query database. The second part then runs the RBnet classifier on these DNS logs once two valid queries have

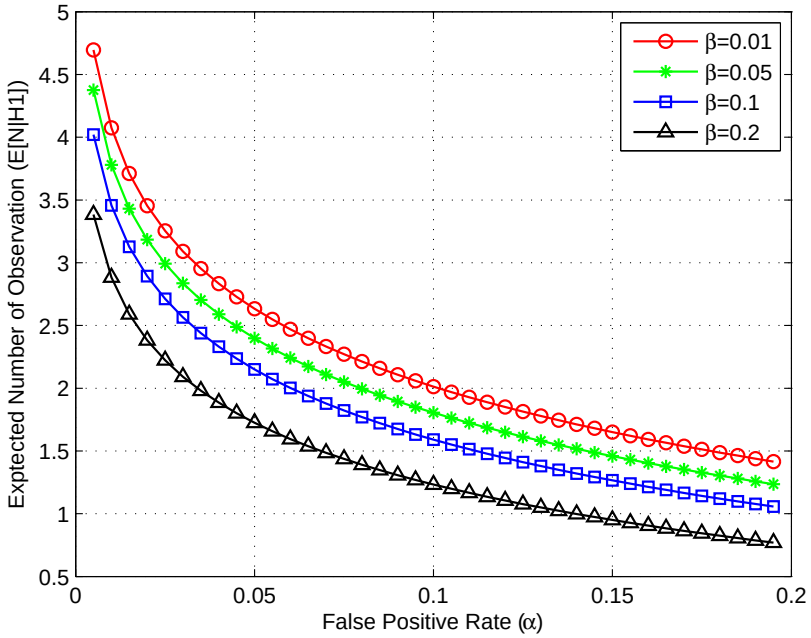


Figure 2.6: Expected number of flows required to determine redirection servers based on inter-flow duration

been obtained. If, based on these two queries, the classifier cannot identify the domain as belonging to a RBnet, it continues to gather DNS queries on the domains until it has accumulated enough for SVM-2 to re-attempt classification. This approach reduces the amount of memory required by the a-DADS and maintains DNS-query logs (in the DNS-query database) for the suspicious domains should manual analysis be required later.

When calculating the suspicion level for a domain, the classifier must read all the domain's data from the DNS-query database, extract the relevant behavioral characteristics from the data, and then use those characteristics in the SVM equation (either SVM-1 or SVM-2). To determine the overhead of the unoptimized, proof-of-concept RBnet classifier, we did run-time performance tests for 150 domains,[5] consisting of

[5]All the domains in the test already had enough DNS-query data in the DNS-query database for the classifier to arrive at a decision. Therefore, these performance measurements don't reflect the time needed to gather the necessary DNS data, only the time taken to process it and calculate suspicion levels.

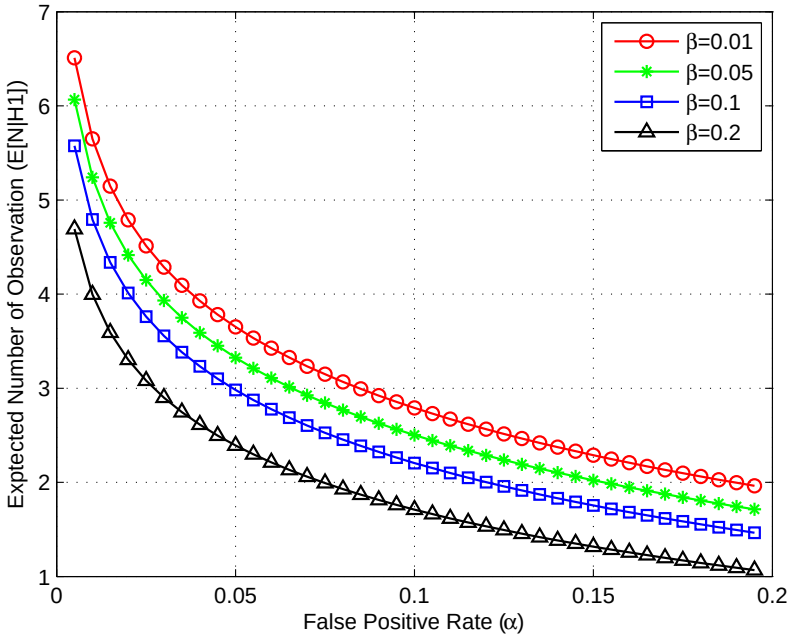


Figure 2.7: Expected number of flows required to determine redirection servers based on flow size

50 randomly-chosen domains from each of the following sets: Aggressive RBnet domains, Stealth RBnet domains, and valid (i.e., benign) domains. We measured the total time it took for the RBnet classifier to classify each domain; this includes such unoptimized operations as reading the data, parsing the data, extracting the characteristics, etc. Therefore, we also measured the amount of time it took for SVM-1 and SVM-2 to calculate the suspicion levels used for classification after the relevant characteristics have been extracted from the DNS-query data. The implementation of SVM-1 and SVM-2 leaves little room for optimization, unlike the rest of the RBnet classifier. We found that the RBnet classifier had an average run-time of 0.644 second per domain. Meanwhile, SVM-1 and SVM-2 had average run-times of 8 and 12 microseconds, respectively, making them suitable for a real-time detection system. Even when unoptimized, the proof-of-concept RBnet classifier takes (on average) less than a second per domain, which is sufficient for fast detection.

2.8.2 Evaluation of RBnet classifier

The a-DADS's RBnet classifier (described in Section 2.6.4) continuously monitored the 91,600+ suspicious domains in the redirection domain database detected by the SSS and the NAS over a period of approximately two months. Utilizing the CDN Filter, the a-DADS was able to determine 4,164 CDN domains (4,506 IPs) based on the reverse DNS name. Using the recursive iteration technique described in Section 2.6.3, this number was increased to 5,005 CDN domains (5,185 IPs), for a 16.8% increase in CDN domains (13.1% increase in IPs). The remaining domains were further filtered for known valid domains, using a technique similar to the CDN Filter. After filtering these domains, the a-DADS continued to monitor the remaining 35,500+ domains, applying SVM-1 and SVM-2.

With just two valid queries, SVM-1 was able to detect 125 Aggressive RBnet domains comprising 3,541 bot IPs. These Aggressive RBnet domains were then removed from the redirection domain database. The a-DADS continued to monitor the remaining domains in the redirection domain database, pruning the list as new CDN Filter data became available. After a week of monitoring, SVM-2 was able to identify an additional 156 Stealth RBnet domains, which comprised 249 IPs. Thus, RB-Seeker managed to isolate a total of 3,790 bot IPs utilized by 281 unique domains for nefarious purposes. Further analysis revealed that 64 of the 125 Aggressive RBnet domains (51.2%) were used as proxies while the remaining 61 (48.8%) were used in a redirection infrastructure. All of the observed Stealth RBnets were involved in redirection infrastructures. SVM-1 experienced one false-positive (FP-rate $< 0.008\%$ for SVM-1), which has been removed from the previously-reported results. The false positive arose from SVM-1 misclassifying a valid *mozilla.org* domain used in a CDN for distributing releases. This single false positive results in the a-DADS's RBnet classifier (SVM-1 and SVM-2) having the low FP-rate of $< 0.004\%$. This false positive can be avoided with the addition of *mozilla.org* to the CDN Filter.

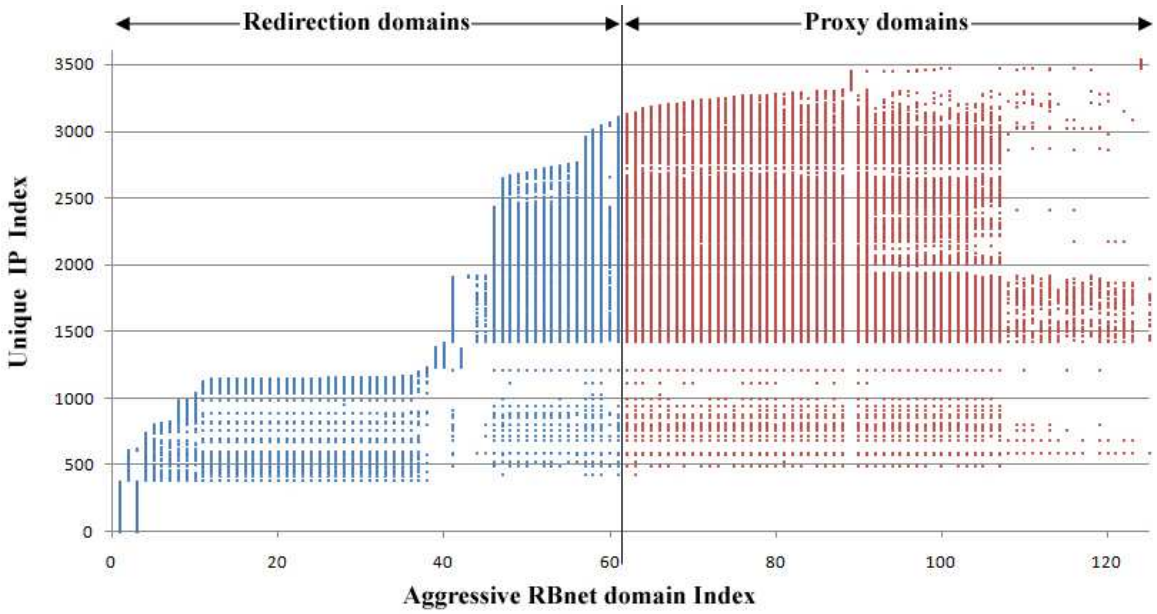


Figure 2.8: Unique IPs (as represented by unique IP index) seen for each detected Aggressive RBnet domain

For comparison, we also implemented the *FFSN detector* described in [39] and ran it on all the domains detected by the NAS and the SSS. The FFSN detector looks at the number of IPs and ASes over two valid queries. It succeeded in correctly identifying 124 of the 125 Aggressive RBnet domains that RB-Seeker detected. It also incurred a false positive, incorrectly classifying the *mozilla.org* domain. Because the FFSN detector only looks at two valid queries, it is unable to distinguish the slower, more stealthy Stealth RBnets from valid domains (such as CDNs). The a-DADS's SVM-1 suffers from this same limitation. However, by utilizing the 2-tier system and monitoring suspicious domains over a longer period of time, SVM-2 is capable of detecting these Stealth RBnets.

2.8.3 Analysis of detected RBnets

Using the 125 Aggressive RBnet domains RB-Seeker detected, we generated the plot in Fig. 2.8, which shows the unique 3,541 IPs (represented by a unique index)

seen for each Aggressive RBnet domain. The graph is divided into two parts along the x-axis; the left half represents domains using redirection while the right half contains domains using only proxies. Likewise, Fig. 2.9 shows a plot of the 156 Stealth RBnet domains and their corresponding 249 unique IPs (also represented by a unique index). The Aggressive RBnets and Stealth RBnets are shown in separate plots because they share no common IPs.

From the figures, it is apparent that while Stealth RBnets have more unique domains than Aggressive RBnets, they utilize vastly fewer IPs. Because Stealth RBnets attempt to mimic the DNS behavior of valid domains (such as CDNs), they must use computers with more stable IPs than Aggressive RBnets. If a Stealth RBnet attempted to use less persistent computers and still keep its DNS behavior "below the radar," the availability of the nefarious content serviced by the RBnet would suffer. It is the unreliable connectivity of the constituent bots in an Aggressive RBnet that necessitates the abundant advertising of IPs. Stealth RBnets, on the other hand, consist of more reliable redirection servers—whether they are compromised machines or legitimate redirection infrastructures that are being exploited. However, this doesn't mean that Stealth RBnets are exactly like valid CDNs. Recall from Fig. 2.5, that over time, Stealth RBnets must continue to supply fresh IPs, either to avoid detection or as a result of being detected. Thus, to differentiate Stealth RBnets from valid CDNs, it is necessary to perform long-term DNS monitoring.

One thing Stealth and Aggressive RBnets do seem to have in common is their use of multiple domains resolving to a similar set of IPs. This can indicate (1) that a single botnet is servicing multiple scams, (2) that a single scam is utilizing multiple domains to evade detection, or (3) that those computers serving as bots are highly susceptible to attacks and have been incorporated into multiple botnets and their scams. It is likely that each of these scenarios plays a role in the observed behavior. However, from Fig. 2.9, it seems that the Stealth RBnets have IPs that are more

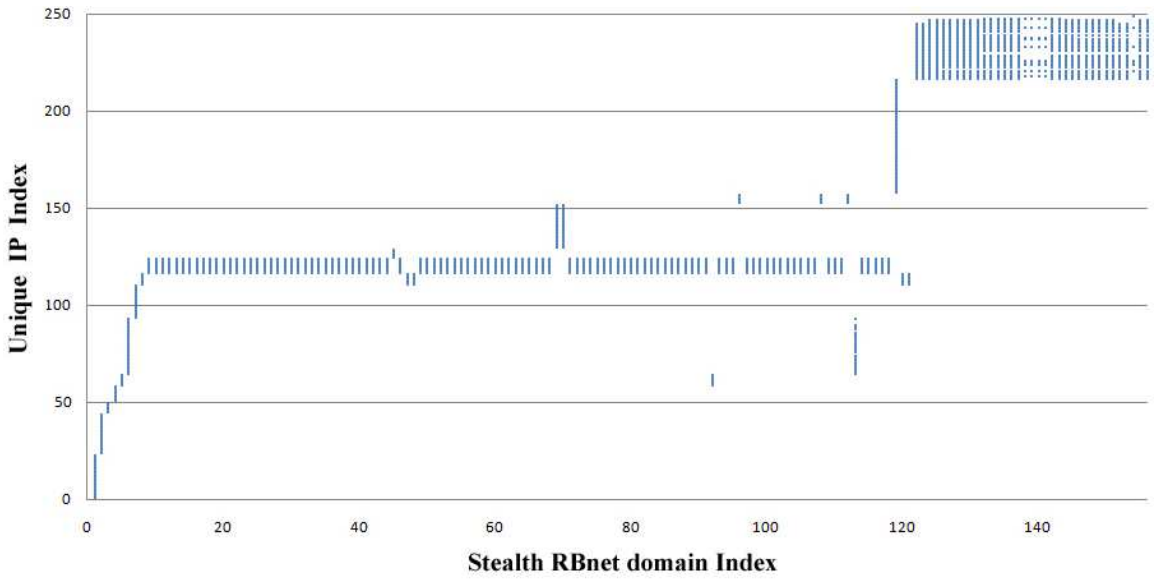


Figure 2.9: Unique IPs (as represented by unique IP index) seen for each detected Stealth RBnet domain

compartmentalized to various domains' indices. This seems to favor the scenario that those domains sharing a set of IPs are aliases for the same scam, while those domains with different sets of IPs are partaking in different scams. For example, there are many domains that seem to share the same IPs with IP indices around 125. These domains are all derivations of *[XXX].bay.livefilestore.com*, where *[XXX]* is some random character string. *Norton Safe Web* identifies *livefilestore.com* as a high risk for Trojans, viruses, worms, spybots, identity theft, and phishing attacks [67], affirming that this Stealth RBnet is involved in numerous malicious activities and employing multiple domain aliases. Looking at Fig. 2.8, we can see that roughly half the Aggressive RBnet domains are using redirection (labeled *redirection domains* in the graph) while the other half are using only proxies (labeled *proxy domains* in the graph). Interestingly, many of the same IPs are shared among both redirection and proxy domains. This lends credence to the scenario that the shared IPs belong to computers which have been compromised by more than one bot, so they are being used in multiple, separate botnet campaigns (i.e., proxy vs. redirection campaigns).

Also, notice from Fig 2.8 that those Aggressive RBnet domains below domain index 40 share only a small set of IPs with the higher domain indices. They also seem to have a large subset of IPs that they share only among themselves. While we would expect a break like this to occur near the division between redirection and proxy domains, this occurs in the middle of the redirection domains. Those redirection domains with a domain index greater than 40 tend to share more IPs with the proxy domains. This seems to indicate that many of these computers are servicing multiple botnet campaigns. While this could be the result of multiple bots compromising the same machines or a botmaster using a single botnet for multiple scams, one thing is certain: by discovering these bots and blocking their IPs, we can potentially mitigate numerous botnet scams.

2.9 Conclusion

In this chapter, we have designed and implemented a prototype system called RB-Seeker for detecting RBnets, which are increasingly used by attackers to redirect unsuspecting victim to web servers hosting nefarious content. Analyzing network traffic and DNS behavior using several statistical correlation and classification techniques, RB-Seeker achieves fast and automatic detection of RBnets, independent of their C&C protocol or structure. First, the SSS and the NAS cooperatively detect redirection domains using various data sources. The NAS identifies redirection domains without the need for expensive and intrusive packet analysis; it leverages temporal/spatial features derived from NetFlow traces that are unique to redirection activities, such as inter-flow duration and flow size. Meanwhile, the SSS identifies domains participating in redirection by inspecting spam emails and following any discovered embedded links they contain. The suspicious domains identified by the SSS and the NAS are then given to the a-DADS, which performs DNS queries on them and identifies malicious domains based on their atypical DNS-query results. Using

real-world network traces to evaluate RB-Seeker, we discovered it is capable of detecting both Aggressive and Stealthy RBnets with low false positives. Since the bots used in RBnets also likely take part in numerous other botnet scams, their fast and automatic detection can protect users from phishing and scam websites while simultaneously encumbering other malicious activities in which they partake. Furthermore, since its data sources are readily available in most enterprise networks, RB-Seeker is incrementally deployable and easily incorporated into existing security systems.

CHAPTER III

Charlatan's Web: The Measurement and Analysis of Global IP-Usage Patterns of Fast-Flux Botnets

3.1 Introduction

Used as a misdirection mechanism for evading detection, botnet-based hosting services often come in tandem with a variety of other criminal scams, constituting an essential portion of botnets' overall operation. For example, spam/phishing campaigns often utilize botnets for misdirection. They begin by using some spamming mechanism (e.g., a hijacked mail server and/or a botnet) to send seemingly interesting phishing emails. Within the phishing emails are innocuously disguised embedded links whose domain names resolve to IP addresses of compromised computers in a botnet. Once victims click the embedded links, they connect to the bots, which then redirect them to—or serve as proxies for—the central host (often called the *mothership*) of the nefarious content. This strategy grants criminals a high level of anonymity while enabling easy and centralized management of the malicious content. However, because botnets are composed primarily of compromised home computers with unreliable connectivity, it is not uncommon for them to unexpectedly go offline (e.g., the computer is turned off or the installed malware is discovered and removed). Botnet-based hosting services, therefore, must be protected against the failure or disruption of

individual bots, ensuring the availability and stability of the hosted service/content. As a result, it is beneficial for bot-based hosting infrastructures to adopt fast-flux (FF) DNS techniques, which frequently change the domain name mappings to different bots' IP addresses. When the victim tries to visit the malicious domain, the DNS server responds with a set of up-to-date, active bot IPs. By recruiting a large pool of IPs and supplying a large set of IPs per query, botmasters can ensure, with high probability, that the malicious domain resolves to at least one valid IP belonging to an online bot. An additional level of control and resilience is attained by giving the domain's IP mappings a short time-to-live (TTL) value. This permits botmasters a quick response when a bot goes offline, replacing its IP with one from the ample supply of online bots. Using this FF technique, botmasters effectively transform their botnets into a global Content Delivery Network (CDN), providing highly available and reliable content-hosting services in spite of node failures. This extends the lifetime of illegal activities (and thus profits) the botnets provide, complicating disruption efforts by introducing an additional layer of misdirection.

Previous research has studied the features of FF botnets and their malicious uses in phishing scams [53] (e.g., Storm Worm and Rock Phish). However, little has been reported on botnets' IP-usage behavior from a global perspective. Because botnets are formed with myriad compromised hosts dispersed around the world, accurate characterization of how botmasters manage this vast number of IPs can only be achieved by collecting and analyzing data from a global viewpoint. In this chapter, we attempt to fill this important gap and explore the global usage patterns of botnets' IP addresses. Our work is unique and different from the previous work in the following ways. First, we build a global query engine called *DIGGER* that monitors complete DNS behavior from 240 geographically-dispersed vantage points for an extended period of time. This provides us with a unique, global view of how different types of domains differ in their IP-usage patterns. Second, we propose effective methods to characterize and

quantify the temporal and spatial IP-usage patterns of FF botnet domains, facilitating the classification and detection of different domain types. This also allows us to reveal several previously-unknown features of FF botnets and uncover new, discreet IP-management strategies currently employed by criminals to evade detection. Third, to help us better analyze the current state of FF botnets and their relative prevalence, we design and implement a multi-level classifier capable of separating different types of malicious and benign domains based on their IP-usage behavior. Applying the classifier on more than three months' worth of data allows us to spot potential trends of FF botnets and demonstrate the wide spectrum of their implementations.

The remainder of this chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 describes our system architecture, defines the terminology we use and explores the global DNS IP-usage patterns for different domain types. Section 3.4 presents our proof-of-concept classifier and its experimental results, and finally, Section 3.5 concludes the chapter.

3.2 Related Work

Among the numerous criminal uses of botnets, their use as hosting or redirection/proxy servers for illegal content and phishing scams provides an ideal platform for financial gain. However, because of the unreliable nature of the bots, more and more botmasters have adopted FF DNS techniques to ensure the availability and stability of their malicious service/content. Fast-flux techniques are characterized by the frequent change of domain name mappings to the IP addresses of different bots. Holz *et al.* [39] studied the characteristics of FF networks and first developed detection algorithms; they extract URL links from spam emails and then identify FF networks based on the number of unique IP addresses returned in DNS queries and the number of unique ASNs to which the IPs belong. Nazario and Holz [53] applied a similar approach to track the use of FF domains and characterize several features of FF bot-

nets, such as member size, lifetime, and top-level domain distribution. Their work demonstrated that continuous data mining of FF DNS records can yield insights into the operations of FF botnets. Despite the increasing awareness of FF botnets to the security community [63], there has been little effort in understanding botnets' global IP-usage patterns for different types of FF domains—in particular, double FF (FFx2) domains. We attempt to fill this important gap by continually monitoring the DNS properties of FF domains from a large number of geographically-dispersed vantage points, allowing us to study their behavior patterns from a global perspective. In addition, since the purpose of using FF botnets is to reliably distribute nefarious content to users despite host failures, their behavior resembles that of traditional CDNs [15] like Akamai and CDNetworks. As a result, we conduct an in-depth, comparative analysis of IP management for both FF botnets and popular CDNs. With this knowledge, we are able to identify features that can accurately distinguish between the different types of FF domains and discern them from other domain types—both benign and malicious.

3.3 Measuring and Analyzing Global IP-Usage Patterns of Domains

In this section, we explore the DNS IP-usage patterns of different malicious and benign domains. First, we describe the deployment of a globally-distributed monitoring system and give an overview of the different domain types we have observed in the gathered data. Then, we discuss various interesting features we've identified that are useful in understanding the inherent operations of the different domains types.

Continent	N. America	Europe	Asia	S. America	TOTAL
DIGGER Nodes	111	95	28	6	240
% of TOTAL	46.25%	39.58%	11.67%	2.50%	

Table 3.1: Global distribution of DIGGER nodes by continent

3.3.1 System Architecture

We created a distributed DNS-query engine called DIGGER, deployed on 240 geographically disparate nodes in the PlanetLab test bed [59]. The nodes were chosen based on the location of the DNS servers they queried, such that DIGGER would issue queries to DNS servers (i.e., digs) in different geographic locations around the world. Table. 3.1 shows the distribution of DIGGER nodes, which is reflective of the overall distribution of available PlanetLab nodes.

A rec	• The address (A) record in a DNS query on a domain. • The IP addresses of the domain's content servers.
NS rec	• The name server (NS) record in a DNS query on a domain. • The domain names (not IP adresses) of the domain's NSes.
NA rec	• The A rec in a DNS query on a domain's *name servers.* • The IP addresses of the domain's NSes.
Reverse DNS lookup	• The result of a DNS-query request for an IP's domain name. (i.e., PTR request) • When performing a DNS query on a domain, we also do a reverse DNS lookup on the domain's A and NA rec IPs.

Table 3.2: Domains' DNS-record data gathered by DIGGER

On each node, DIGGER performed DNS queries on a set of domains to gather the information shown in Table 3.2. Based on a domain's most recently returned DNS results, DIGGER continued to dig active domains periodically based on their observed TTL, ensuring fresh DNS-query results. Domains determined to be offline were intermittently dug every 24 hours, so that DIGGER could discover if they came back online. The set of suspicious domains monitored by DIGGER was compiled from multiple sources, including online repositories of phishing [14] and malware [13]

websites; additional suspicious domains were extracted from URL links embedded in spam emails, which were collected from a spam relay trap and recent additions to online spam repositories [37]. As a result, the domains probed by DIGGER tend to be malicious in nature, which is desirable for our purpose of studying malicious FF domains. DIGGER was deployed and gathered global DNS-usage patterns for over 3.5 months in early 2009 on 5,171 active domains. Analysis of this data has revealed several distinct types of IP-usage patterns employed by malicious and benign domains. Next, we will describe these domain types whose differentiating features will be explored throughout this paper.

3.3.2 Domain Types

Before delving into the details of different domain features, we present the reader the following high-level overview of the domain-type nomenclature. To provide an intuitive view of these domain types, we have plotted the global IP usage—as seen from the DNS queries—for some representative domains in Figs. 3.1 and 3.2. In this figure, the *Time* axis represents the time since DIGGER started monitoring the domain; *Node Index* represents the DIGGER node that the IP was observed on, with positive values indicating an A-rec IP and negative values an NA-rec IP; *IP Index* is a unique index incrementally assigned to each newly-observed IP.

FF domains (Fig. 3.1 (a)-(c)) are malicious domains utilizing a fast-flux (FF) DNS-advertisement strategy, typically built atop botnets. Because bots may unexpectedly go offline, FF domains advertise numerous IPs in their DNS-query results, helping ensure some of the IPs belong to a functional bot. The TTL of the IPs used by FF domains tend to be relatively short; this permits botmasters a finer level of control in replacing IPs advertised to DNS servers, increasing the availability of an online bot and access to the malicious payload. It is this excessive number of constantly-changing IP addresses that qualifies a domain as "fluxy", and the domain is considered a FF

domain. Domains exhibiting FF behavior in only a *single* record type (i.e., A rec or NA rec) are considered **FFx1 domains** (single fast flux). More specifically, FFx1 domains that are fluxy in their A recs (i.e., content servers) are termed **FFx1_Arec domains** (Fig. 3.1 (a)), while those that are fluxy in their NA recs (i.e., name servers) are termed **FFx1_NArec domains** (Fig. 3.1 (b)); FFx1_NArec domains are able to evade current detection strategies that focus on A recs by migrating their fluxy behavior to their NA recs, where it is less likely to be noticed. When FF domains are fluxy in *both* their A and NA recs, they are considered double fast flux, or **FFx2 domains** (Fig. 3.1 (c)).

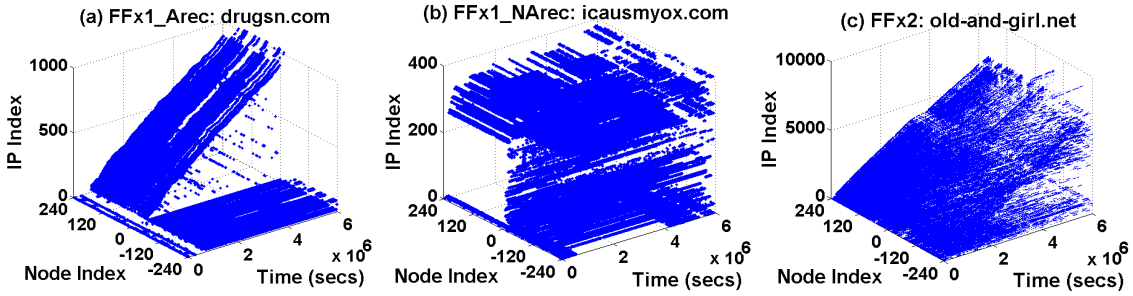


Figure 3.1: Global IP-usage patterns (in DNS-query results) for some examples of the FF domain types

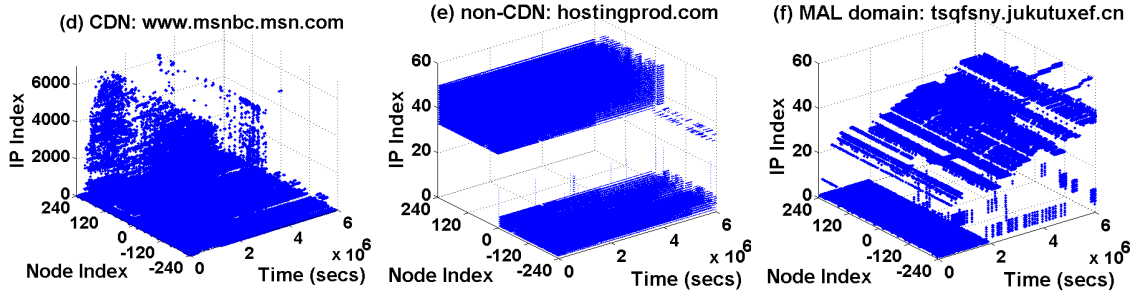


Figure 3.2: Global IP-usage patterns (in DNS-query results) for some examples of the non-FF domain types

CDN domains (Fig. 3.2 (d)) are valid, benign domains that uses a Content Delivery Network, such as Akamai, to improve the delivery of their content. CDNs—consisting of a system of computers networked together for the purposes of improving the performance and scalability of content distribution—produce DNS-query results

resembling those of malicious FF domains: numerous, changing IPs per query with short TTL values. This affinity is a consequence of their similar goal to provide reliable content delivery despite node failure as well as their shared assumption that any node can temporarily or permanently fail at any time. However, CDN domains demonstrate geographic awareness (i.e., IPs geographically close to a DNS server will be advertised with higher probability at that server) and load balancing (i.e., techniques improving performance and scalability not observed in FF domains).

Non-CDN domains (Fig. 3.2 (e)) are valid, benign domains that *don't* use a CDN for delivery of their content. Typically, non-CDN domains use a few stable content servers and name servers (NSes) during the entire monitored period.

MAL domains (Fig. 3.2 (f)) are domains that aren't fluxy enough to be considered FF domains, nor benign enough to be considered non-CDN domains. Their DNS behavior demonstrates potentially suspicious behavior often attributed with malicious domains. They tend to recruit more IPs than a non-CDN, but not nearly as many as a FF domain. For example, during a monitoring period of a few months, a FF domain typically advertises thousands of different IPs. A MAL domain, on the other hand, will advertise perhaps a total of 20–30 IPs—roughly one or two new IPs every few days. MAL domains will tend to slowly add more IPs because they will slowly lose some as their malicious activities are detected and their IPs are blocked.[1] The IPs used by MAL domains may consist of bots or valid servers being used for malicious means.

Having introduced the nomenclature we adopted to describe different domain types, we now present several interesting features of their IP-usage patterns we discovered through the analysis of the globally collected data. For our subsequent analysis, we manually identified 15 domains of each type (i.e., 90 domains total), allowing

[1]Notice, websites hosted on home computers with dynamic IP addresses could be considered MAL domains by our definition. However, we consider this acceptable since most valid websites are not hosted on home computers, causing those that are to be inherently suspicious.

us to better compare and identify distinguishing features between them.

3.3.3 Number of Unique IP Addresses per Node

The first feature we examine is the number of unique IPs seen across the DIGGER nodes over time. Fig. 3.3 and 3.4 show the CDFs of the number of unique A-rec and NA-rec IPs observed by our 240 DIGGER nodes during the ≈3.5 month monitoring period for some representative domains. MAL domains have been omitted in the plots due to their similarity to non-CDN domains. Our empirical data reveals that non-CDN and FFx1_NArec domains (whose A recs behave like a non-CDN) use a small set of stable content servers. For example, in Fig. 3.3, neither the non-CDN nor the FFx1_NArec domains contain more than 18 unique A-rec IPs per node. CDN domains are sometimes found to demonstrate a larger number of unique A-rec IPs on some nodes, though the number of nodes is considerably fewer than observed for FFX1_Arec and FFx2 domains. For example, for the CDN in Fig. 3.3, ≈2% of the DIGGER nodes observed more than 100 unique A-rec IPs. On the other hand, FFx1_Arec and FFx2 domains clearly possess a much greater number of unique A-rec IPs on a larger percentage of nodes—a direct consequence of the bots' unreliable connectivity. For the FFx1_Arec domain in Fig. 3.3, more than 35% of nodes detected over 200 unique IPs, and a few observed over 700. The numbers observed for the FFx2 domain are even higher, with over ≈63% of the nodes observing over 200 unique IPs, ≈43% more than 500, and several more than 2,500.

While the number of unique A-rec IPs per node appears a promising distinguishing feature, our data implies that this is not the case for the average number of unique *NA-rec* IPs. From Fig. 3.4 it is apparent that CDN and FFx2 domains possess many more unique NA-rec IPs per node than the other domain types. On average, FFx2 and CDN domains advertise 999 and 727 unique IPs on a single node, respectively. It seems that, over time, CDNs can advertise numerous name server (NS) IPs. This

behavior might arise from the CDN trying to ensure the availability of its NSes, affording it better control when performing load balancing.

To better understand how the number of unique IPs per node can differentiate between the different domain types, let us quantify the feature as follows:

F1 Let P_i = number of unique IPs on node i, and let N = number of nodes (of the 240 total) where the number of unique IPs ≥ 1. Then, the average number of unique IPs per node (F1) is computed as:

$$\text{F1} = \frac{\sum_{i=1}^{N} P_i}{N} \tag{3.1}$$

Using Eq. (3.1), we calculate F1 over a typical week for the 90 manually identified domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.5. Most malicious FF domains aren't online during the entire ≈3.5-month monitoring period, eventually becoming detected and blocked; therefore, calculating F1 over a single week when the domains are active grants us a more standardized representation for comparison. Figure 3.5 affirms that FFx1_Arec and FFx2 domains have a considerably higher average number of unique A-rec IPs per node than the other domain types. Additionally, we find that within these FF domain types, the actual average number of A-rec IPs per node is highly variable, indicating that there is a wide spectrum in the amount of observable fluxiness demonstrated within a single week. Furthermore, we find that CDN domains actually demonstrate a considerably higher value for F1 in their NA rec than the other domain types, including FFx2 domains. Interestingly, while the FFx1_NArec domains do not use as many NA-rec IPs per node on average as their FFx2 counterparts, they are much more consistent in their advertising strategies, and they tend, on average, to use more IPs per node than their non-CDN counterparts (i.e., FFx1_Arec, MAL, and non-CDN). When looking at the A+NA-rec IPs together as a single IP pool, we find that the CDN, FFx1_Arec, and FFx2 domains distinguish

themselves from the other domain types. Consequently, while F1 seems only partially successful in its ability to identify FF domains from the other domain types, it could prove a powerful tool in distinguishing CDN/FF domains from non-CDN/MAL domains, allowing us to rely on other features to separate CDN and FF domains.

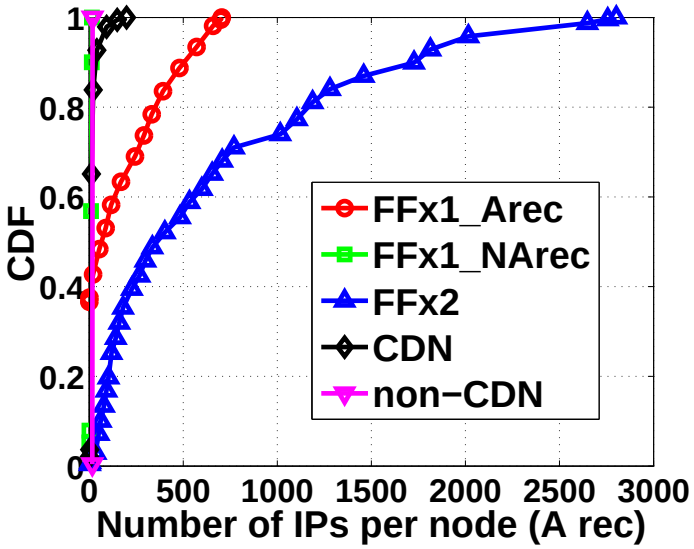


Figure 3.3: CDF of number of unique A-rec IPs per DIGGER node (all weeks)

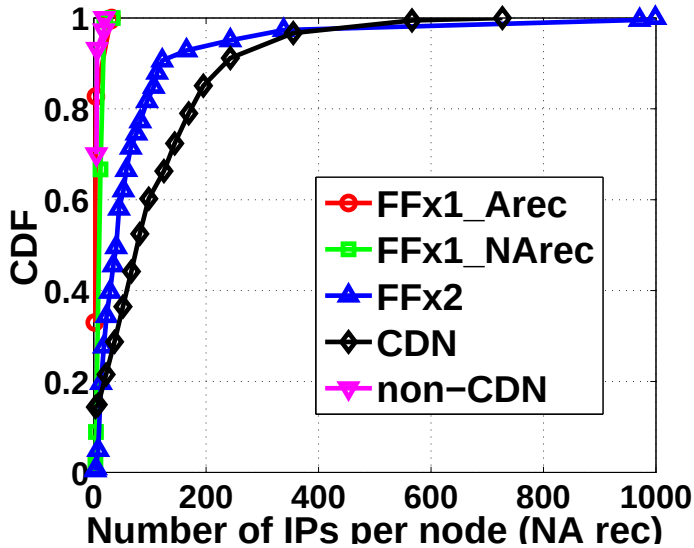


Figure 3.4: CDF of number of unique NA-rec IPs per DIGGER node (all weeks)

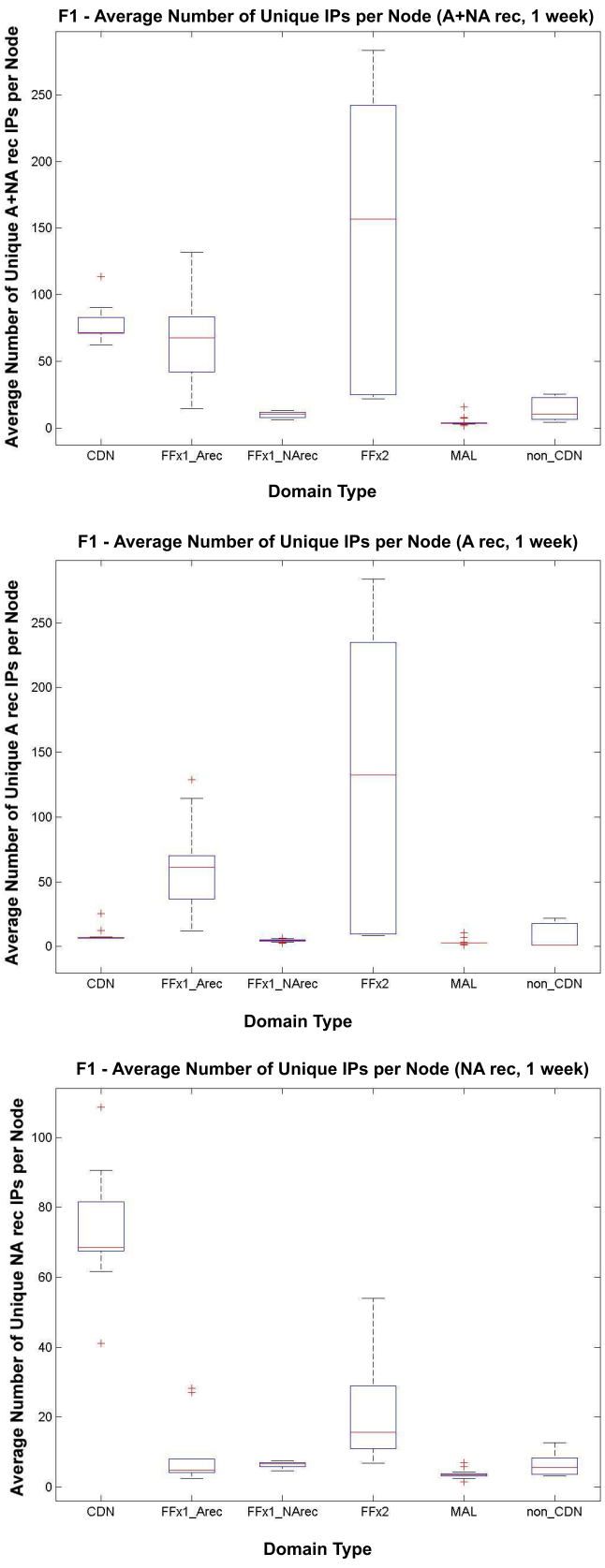


Figure 3.5: Average Number of Unique IPs per DIGGER Node (1 week)

3.3.4 Number of Nodes per IP Address

Since the number of unique IPs per node proved a promising feature for differentiation, we decided to look at the relationship from the inverse perspective: for individual IP addresses, how many different nodes (i.e., DNS servers) were the IP addresses observed on. We restate this as the *number of nodes per IP address*, and it attempts to capture some spatial aspect of the various domains' DNS advertisement strategies.

First, we will examine the CDF for the number of nodes per A-rec and NA-rec IP for some representative domain types, shown in Fig. 3.6. From the figure, we can clearly see that FFx2 and FFx1_Arec domains exhibit remarkably similar trends in their A recs, separate from those of the other domain types. Since the non-CDN domain advertises its small set of stable A-rec IPs with every DNS server around the globe, we observe nearly all the IPs on every node and, therefore, a large number of nodes per IP. The FFx1_NArec domain behaves similarly. However, since some of its A-rec IPs might belong to bots (or otherwise unstable content servers), some of its IPs may not appear on all nodes. As a consequence of its location-aware DNS-advertisement strategy (i.e., advertising IPs geographically close to the queried DNS server), many of the CDN domain's IPs will only be advertised to a small set of nearby DNS servers, generally keeping its average number of nodes per IP small and setting it apart from the other domain types. Likewise, the FFx2 and FFx1_Arec domains fall somewhere in between due to their necessity-based DNS-advertisement strategy (i.e., limited to advertising whichever bots happen to be currently online and available to serve content). It may be the case that bots with unstable connectivity only get advertised to a handful of DNS servers before they go offline. If a bot only intermittently loses connectivity, its IP may eventually propagate to more DNS servers, increasing its nodes-per-IP count. However, if the bot permanently disconnects from the botnet, its nodes-per-IP count will remain stunted, decreasing the overall average

nodes-per-IP.

Contrary to the A-rec IPs, the different domain types all demonstrate similar behavior in relation to their NA-rec IPs. We find that the non-CDN and FFx1_Arec domains exhibit very similar behavior since, like non-CDN domains, FFx1_Arec domains utilize traditional name servers for their NA-rec IPs. Furthermore, we have found that FF domains tend to use some of their more stable and consistently online bots for their NA-rec IPs. In this way, if the domain's content servers are all offline, there is a higher probability that bots serving as the name servers can be reached, allowing them to provide fresh, online bot IPs for the domains A rec. While they use some more stable bots for their NA-rec IPs, FF domains still augment this with their normal, less-stable bots. Thus, we find a small portion of stable bot IPs appearing on many nodes, while the less stable bots go offline before appearing on too many nodes. Consequently, FFx2 and FFx1_NArec domains demonstrate similar behavior in their NA recs as CDNs, though for different reasons. CDN domains tend to use location-aware advertising strategies, and so their behavior results from only advertising certain name servers (i.e., NA-rec IPs) to nearby DNS servers. In any case, the different domain types NA-rec behavior appears too similar to make this a useful differentiating metric.

To better understand if the number of nodes per IP can be useful in differentiating between the various domain types, let us quantify the feature as follows:

F2 Let N_i = number of different nodes a unique IP, i, was seen on, and let P_T = total number of unique IPs seen over all nodes. Then, the average number of nodes per IP (F2) is computed as:

$$\text{F2} = \frac{\sum_{i=1}^{P_T} N_i}{P_T} \tag{3.2}$$

Using Eq. (3.2), we calculate F2 over a typical week for the 90 manually identified

domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.7. The figure confirms that the NA-rec IP behavior is too similar among the different domain types to be useful in classification. It has also revealed that FFx2 and FFx1_Arec domains do not always exhibit unique behavior from the other domain types, and sometimes appear similar to non-CDN/MAL domains. From the A-rec box plot in Fig. 3.7, we find that the FFx2 and FFx1_Arec domains demonstrate different behavior from the CDN domains. They also tend to exhibit different behavior than the other domain types for their A-rec IPs, as was demonstrated in Fig. 3.6; however, the strength in differentiation is not as strong as with the CDN domains due to their unique location-aware advertisement strategy, and there is some overlap in their F2 values. These somewhat nebulous results imply that the number of nodes per IP address may not serve as a useful feature in differentiation.

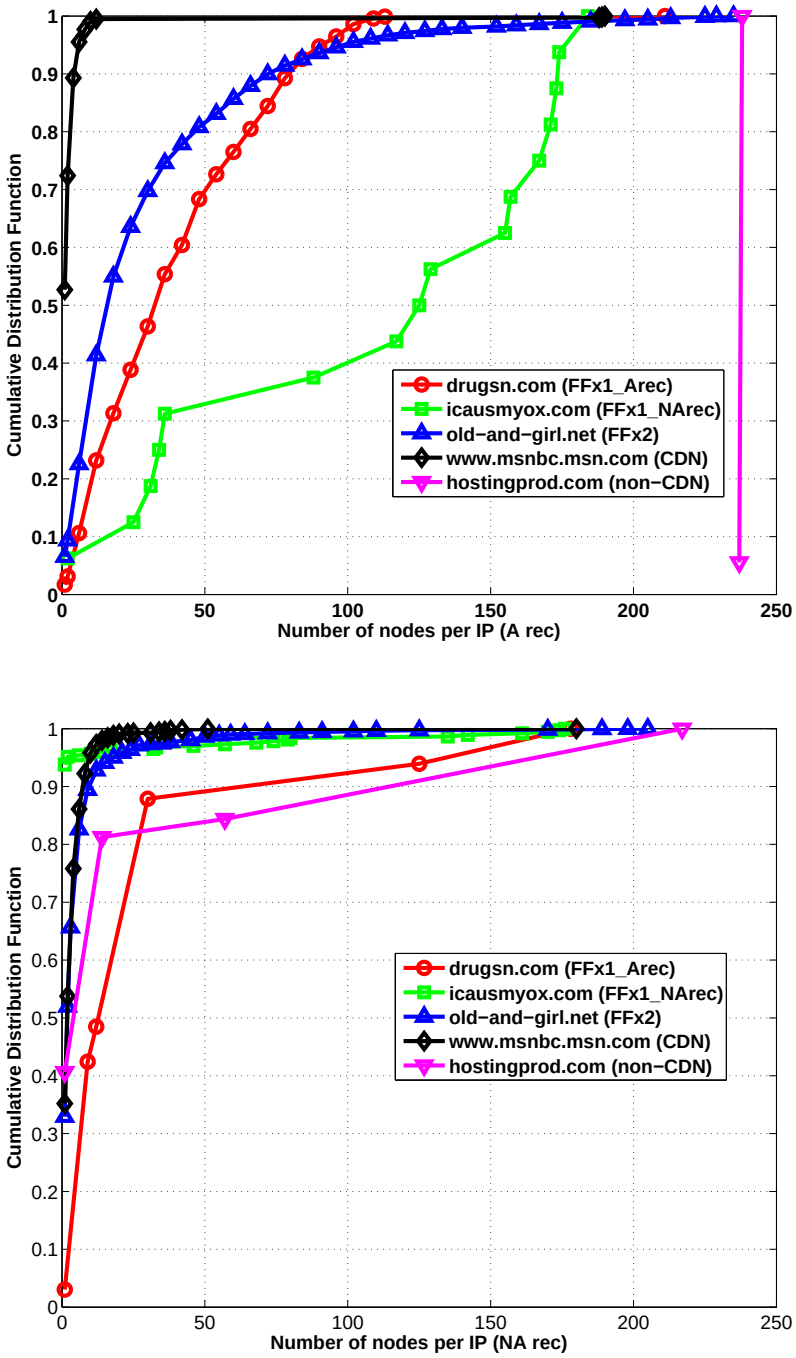


Figure 3.6: CDF of number of nodes per IP (all weeks)

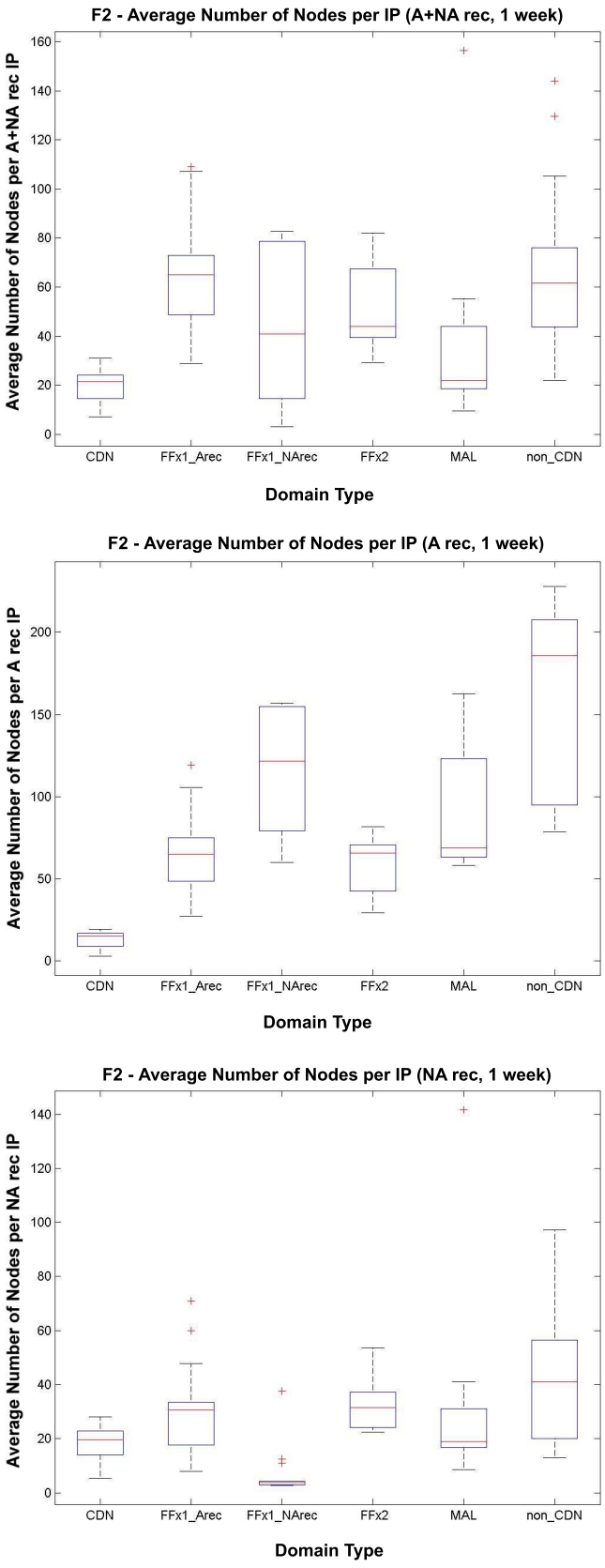


Figure 3.7: Average number nodes per IP (1 week)

3.3.5 Overlap between IPs of A and NA Records

While analyzing our data, it quickly became apparent that FF domains tend to exhibit some IP overlap. We observed IPs advertised for a domain's A rec reappearing in the same domain's NA rec. Furthermore, when DIGGER performed DNS digs on the domain's NSes, the same IP would often be returned for different NSes. We discovered that the malicious domains were not only reusing their available IP pool for both A and NA recs, but were also returning IPs from the same IP pool regardless of which NS was queried, resulting in different NSes with identical IPs—a trend not witnessed for benign domains.

Table 3.3 shows the total number of A-rec, NA-rec, and overlap IPs for some representative domains from each domain type. This overlap phenomenon was much more prevalent in FFx2 domains than either type of FFx1; we never observed it in valid, benign domains. The FFx1 domains almost entirely use valid IPs for one record type and the IPs of compromised computers for the other. While the representative MAL domains have a small number of total unique IPs (like non-CDN domains), their IP overlap is exceptionally high, with almost all of their A-rec IPs also used for their NA recs, setting them apart from valid domains. The IP overlap we empirically observed evinces that valid domains use separate machines for their content and name servers to prevent a single point of failure. FF and MAL domains, on the other hand, attempt to make the most of their limited resources, reusing IPs for both the A and NA records.

So that we can draw more accurate comparisons between the different domain types, let us formally quantify the feature of IP overlap as follows:

F3 represents the percentage of unique IPs that overlap between the A and NA recs. Thus, if all the IPs from one record type are also used for the other record type, there will be a 100% IP overlap. For a given domain across all nodes, let P_A be the set of

Domain Type	Domain	A rec	NA rec	Overlap
FFx1_Arec	drugsn.com	932	33	0
	www.couldchoose.com	486	37	5
FFx1_NArec	icausmyox.com	16	370	1
FFx2	old-and-girl.com	5,227	3,047	879
	mountainready.com	4,060	2,219	2,144
MAL	duelready.com	16	32	15
	tsqfsny.jukutuxef.cn	23	42	20
CDN	www.msnbc.msn.com	1,160	5,412	0
non-CDN	hostingprod.com	18	32	0

Table 3.3: Total A-rec, NA-rec, & overlap IPs for the different domain types

unique A-rec IPs and P_{NA} be the set of unique NA-rec IPs. Then, F3 is calculated as:

$$\text{F3} = \frac{|P_A \bigcap P_{NA}|}{\min\{|P_A|, |P_{NA}|\}} \tag{3.3}$$

Using Eq. (3.3), we calculate F3 over a representative week for the 90 manually identified domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.8. As can be seen from the figure, CDN and non-CDN domains demonstrate no observable IP overlap. While MAL domains seem to exhibit the highest rate of IP overlap at nearly 100%, it should be noted that there are some MAL domains which demonstrate no IP overlap; those possessing no IP overlap will be difficult to discern from non-CDN domains if not monitored for a significant duration of time. The FF domains tend to demonstrate anywhere from no IP overlap to almost 100% overlap. The FFx1_NArec domains seem to demonstrate this behavior the least, with about half of them containing no IP overlap and the rest all containing less than 20% overlap. While about half of the FFx1_Arec domains also possess no IP overlap, the other half has considerably more overlapping IPs than the FFx1_NArec domains, evenly spread from 0 to nearly 100% overlap. This indicates that some FFx1_Arec domains are augmenting their name servers with bots from their A rec, yet not so much as to make their NA recs appear fluxy. From the figure, it is clear that the FFx2 domains consistently exhibit

the most IP overlap among the FF domain types; only a few outliers contain no IP overlap, while the majority demonstrate between 15% and nearly 60% overlap. Since FFx2 domains make extensive use of bots in both their A and NA recs, this high and consistent overlap observed is justified. Furthermore, while some possess exceptionally high rates of overlap approaching 100%, most do not, indicating that FFx2 domains are reserving some of their bots for a particular record type. This trend results from a combination of advertising their more stable bots exclusively in NA recs and their unstable bots exclusively in A recs. Name servers are not actually visited as frequently as content servers and, consequently, possess a lower risk of detection and blocking. By keeping their most stable bots (i.e., those most consistently online) in the NA rec, botmasters decrease their risk of detection and mitigation, further ensuring they are more reachable than their other, less reliable bots. In addition, should all the advertised A-rec bots go offline or become blocked, it is imperative to the botmaster that a NA-rec bot be reachable to provide fresh bot IPs for the A rec. Therefore, it behooves botmasters to protect a select set of stable bots for use exclusively as name servers. Clearly, IP overlap should prove a useful feature for helping identify malicious FF and MAL domains from the other benign domain types.

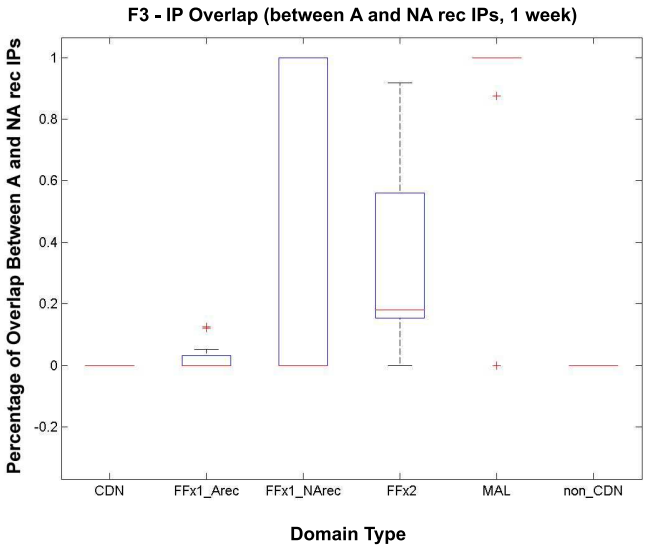


Figure 3.8: IP Overlap between A-rec and NA-rec IPs (1 week)

3.3.6 Continental Distribution of IPs

Next, we examine how the various domain types differ in their IP distribution (i.e., where the IPs returned in DNS queries are geographically located). We examine the IP location based on continent rather than country because the close proximity of European countries made a country-based resolution too fine-grained. In particular, we examined two features: 1) *percentage of IPs from the wrong continent*, i.e., what percentage of IPs returned in DNS queries are located in a different continent than the queried DNS server; 2) *continental IP distribution*, i.e., from the perspective of each continent containing queried DNS servers, what percentage of IPs returned are located in each continent.

Figure 3.9 shows the percentage of A-rec and NA-rec IPs from the wrong continent for some representative domains. From the figure, it is evident that the CDN domain has a considerably smaller proportion of IPs from the wrong continent than the other domain types. The few CDN IPs from the wrong continent are due to load balancing. For example, to distribute load when traffic volume is high in Asia, CDNs may advertise some European IPs to Asian DNS servers, resulting in a small

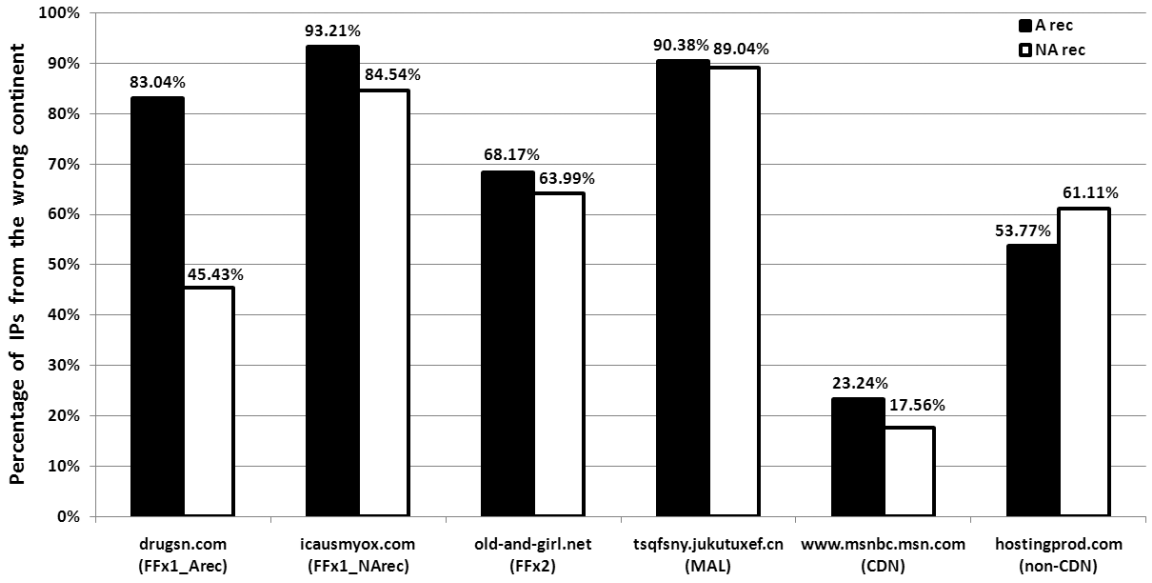


Figure 3.9: Percentage of IPs from wrong continent

percentage of IPs from the wrong continent.

Insight into continental IP distribution can be found in Fig. 3.10 for some representative domains. For brevity, we have not plotted any FFx1 domains, since their results are a subset of the FFx2 domain type; likewise, we have omitted plots for a MAL domain since their distribution is functionally similar to non-CDN domains. In Fig. 3.10, the bars represent the continental IP distribution from different perspectives. In each domain's plot, the first bar represents the continental IP distribution from a global perspective, while the other bars are from the perspective of the different continents where we deployed DIGGER nodes. For example, the bar labeled "Asia" under *old-and-girl.com* indicates the percentage of IPs located in each continent base on queries to Asian DNS servers.

From Fig. 3.10, we notice that the continental IP distribution for CDN domains varies greatly across the different continents, clearly revealing the *location-aware DNS advertisement* employed by CDNs. The DNS-query results for CDN domains often contain a majority of IPs located near the query issuer, providing fast, reliable ser-

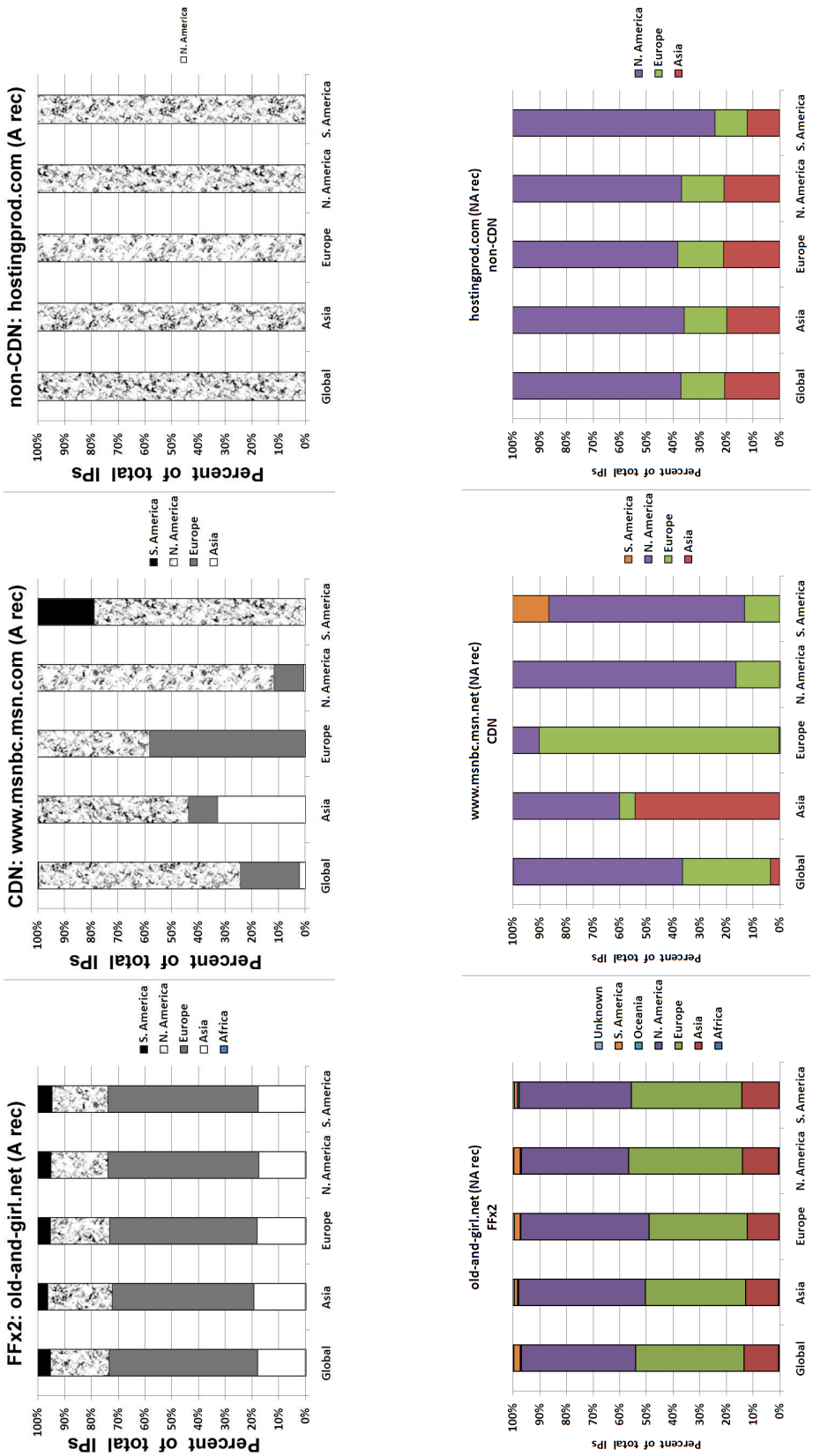


Figure 3.10: Percentage of total A-rec and NA-rec IPs seen from each continent by DIGGER nodes globally and in each continent

vices and quicker content delivery to end users by reducing the data's travel distance. Consequently, CDNs demonstrate a smaller percentage of IPs from the wrong continent and a larger variance in continental IP distribution than other domain types. On the other hand, we find that MAL and non-CDN domains operate in a similar manner. They indiscriminately advertise their small pool of stable server IPs around the world nearly simultaneously. This causes the continental IP distribution at each continent to be identical, and the percentage of IPs from the wrong continent reflects the global distribution of our DIGGER nodes.[2] Finally, our analysis suggests that FF domains adopt an advertisement strategy dictated by the unstable nature of their constituent bots, which we term *necessity-based DNS advertisement*. Since bots can go offline at any moment, FF domains must rely on whichever bots are currently available and advertise their IPs to DNS servers around the globe as necessity dictates, regardless of geographic location. This results in a large percentage of IPs from the wrong continent and a fairly consistent continental IP distribution across continents.

Let us quantify these distribution features so that we may better compare them between domain types as follows:

F4 Using an online database [29], we were able to determine the country of origin for most IPs observed by DIGGER. For those IPs not present in the database, we were able to perform a "who is" lookup and determine most of their countries of origin. The few remaining IPs whose locations couldn't be determined were labeled as "unknown". Thus, for nearly all IPs monitored by DIGGER, we could determine which continent the IP was located on: N. America, S. America, Europe, Asia, Africa, Oceania, Antarctica, and—very rarely—unknown. Let W_i = number of unique IPs on node i that are located in a different (i.e., wrong) continent than node i. Let

[2]Figure 3.10 shows that for the non-CDN domain *hostingprod.com*, almost all of the A-rec IPs are from N. America. Because about 45% of our DIGGER nodes are located in N. America (Table 3.1), we find that 53.77% of *hostingprod.com*'s IPs are from the wrong continent (Fig. 3.9), approximately the same percentage as DIGGER nodes *not* in N. America.

P_i = the total number of unique IPs on node i. Then, the percentage of IPs from the wrong continent (F4) is computed as:

$$\text{F4} = \frac{\sum_{i=1}^{N} W_i}{\sum_{i=1}^{N} P_i}. \tag{3.4}$$

F5 We want to determine the *average* continental IP distribution across all nodes from a given continent. To obtain this, we grouped the nodes together based on the continent they are located in. Then, we examined each group of nodes, tallying the number of unique IPs (per node) seen from each continent. If, for example, an IP appears on more than one node from a given continent, it will be counted once for each node it appears on. Calculating a continent's continental IP distribution in this way is more robust to misbehaving or abnormal nodes and better reflects the continental IP distribution of the majority of nodes from a given continent.

Since CDN domains differ from the other domain types due to their location-aware DNS advertisement strategy, their continental IP distribution will be biased in favor of the queried node's continent. Contrarily, the other domain types will demonstrate nearly identical continental IP distributions regardless of the queried node's location (see Fig. 3.10). Therefore, we want to quantify how similar this distribution appears between continents, enabling us to discern CDNs from the other domain types.

Let the continents N. America, S. America, Europe, Asia, Africa, Oceania, Antarctic and "unknown" be represented by the numbers 1–8, respectively. Then, n_i = number of nodes on continent i, for $1 \leq i \leq 4$ (continents with DIGGER nodes). For node j, let $\hat{a}^j$ be a vector representing the number of unique IPs seen from each continent. Thus, $\hat{a}_i^j$ is the number of unique IPs from continent i that were seen on node j. Then, for each continent i with DIGGER nodes, where $1 \leq i \leq 4$, we calculate $\hat{A}^i$ as shown in Eq. (3.5). We calculate the cosine similarity (shown in Eq. (3.6)) between every possible pair of vectors $\hat{A}^i$, for $1 \leq i \leq 4$, and then take the average,

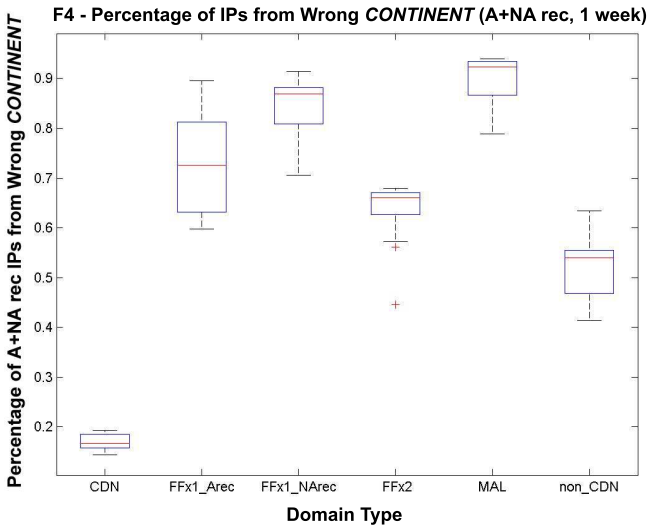
producing the IP continental distribution's average cosine similarity (F5). The closer this value is to 1, the more similar the continental IP distributions appear on each continent and the less likely the domain is a CDN domain.

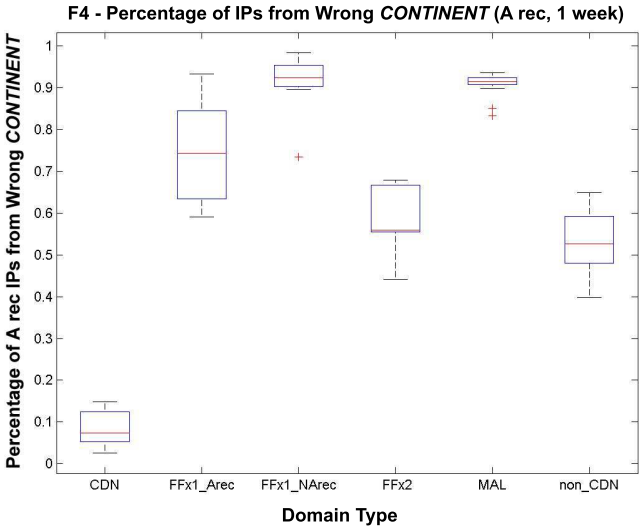
$$\hat{A}^i = \sum_{j=1}^{n_i} \hat{a}^j \qquad (3.5)$$

$$\text{Similarity}(\hat{X}, \hat{Y}) = cos\theta = \frac{\hat{X} \bullet \hat{Y}}{\| \hat{X} \| \| \hat{Y} \|} \qquad (3.6)$$

Using Eq. (3.4), we calculate F4 over a typical week for the 90 manually identified domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.11. Likewise, we produce a similar plot for F5 using Eqs. (3.5) and (3.6), shown in Fig. 3.13. First, from Fig. 3.11, we find that for both the A-rec and NA-rec IPs, the percentage of IPs from the wrong continent serves as a powerful differentiating feature between CDN domains and the other domain types. CDN domains seldom to advertise more than 20% of their IPs from from the wrong continent, while the other domain types never advertise less than 35% from the wrong continent, and most advertise more than 40%. For comparison purposes, we used a variation of Eq. (3.4) to calculate and plot the percentage of IPs from the wrong *country* in the same manner, shown in Fig. 3.12. The figure affirms our previous claim that the close proximity of countries—especially in the European Union—results in a too fine-grained resolution to be particularly useful in differentiation. While there is still some separation between CDN domains and the other domain types in the A rec, it is lost in the NA rec. Even the separation present in the A rec is not as distinct as when using continents to generate F4. Next, examining Fig. 3.13, we discover that feature F5 exhibits an even stronger separation between CDN domains and the other domain types than feature F4. Despite CDN domains' complex use of load-balancing and location-aware advertisement strategies, they seem to invariably produce a different continental distribution fingerprint for each monitored continent, resulting in a considerably lower average cosine similarity than the other domain types. Interestingly, in terms of their A-rec IPs, the other domain

types seem to advertise the same set of IPs nearly everywhere, which accounts for their average cosine similarity being nearly 1. These findings indicate that the percentage of IPs from the wrong continent and the variance of the continental IP distribution are useful features for distinguishing CDN domains from the other domain types.





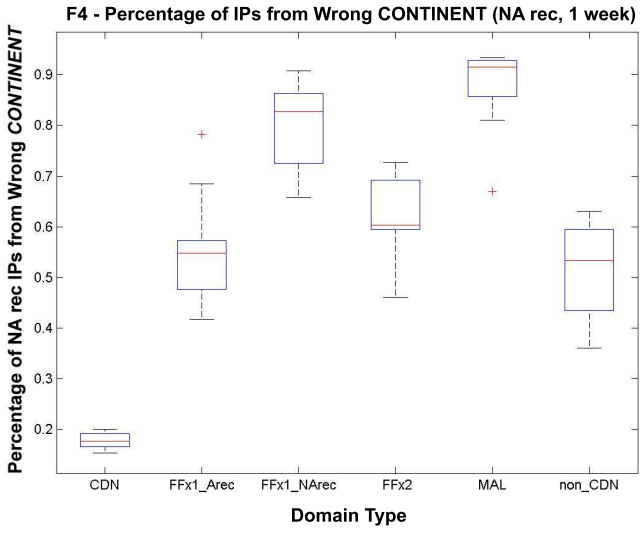
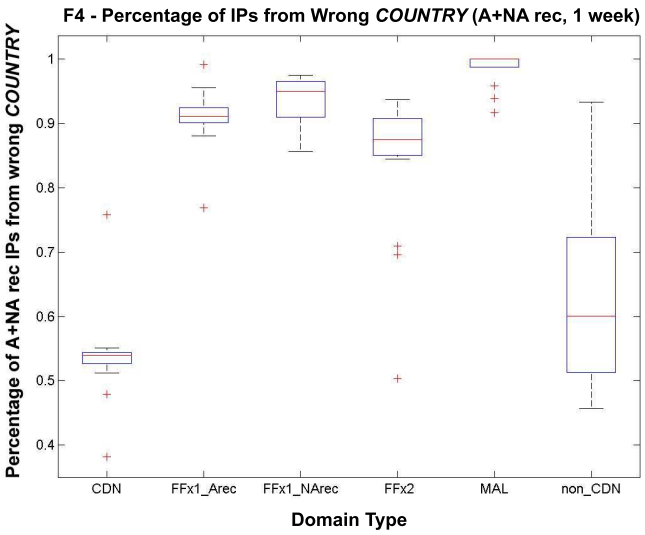
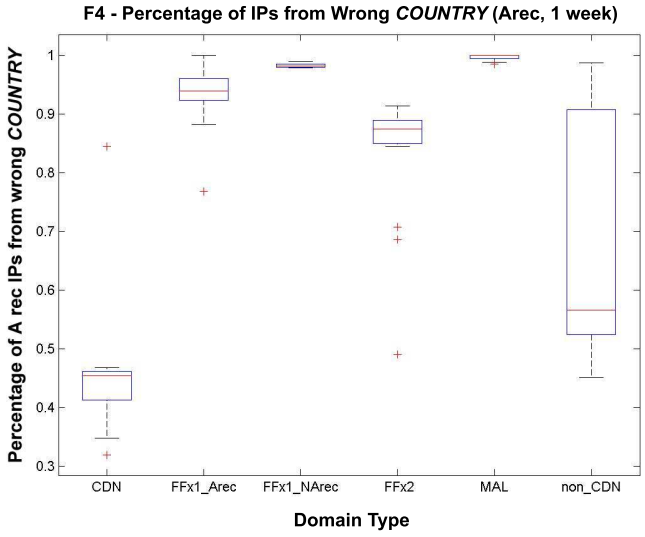


Figure 3.11: Percentage of IPs from wrong CONTINENT (1 week)





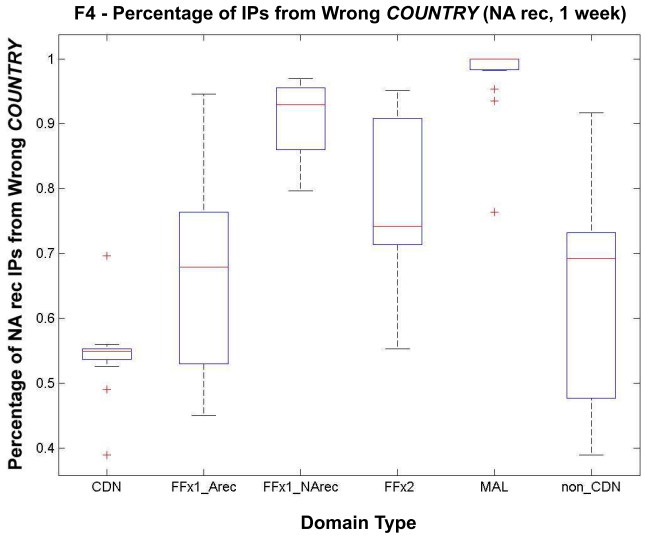


Figure 3.12: Percentage of IPs from wrong COUNTRY (1 week)

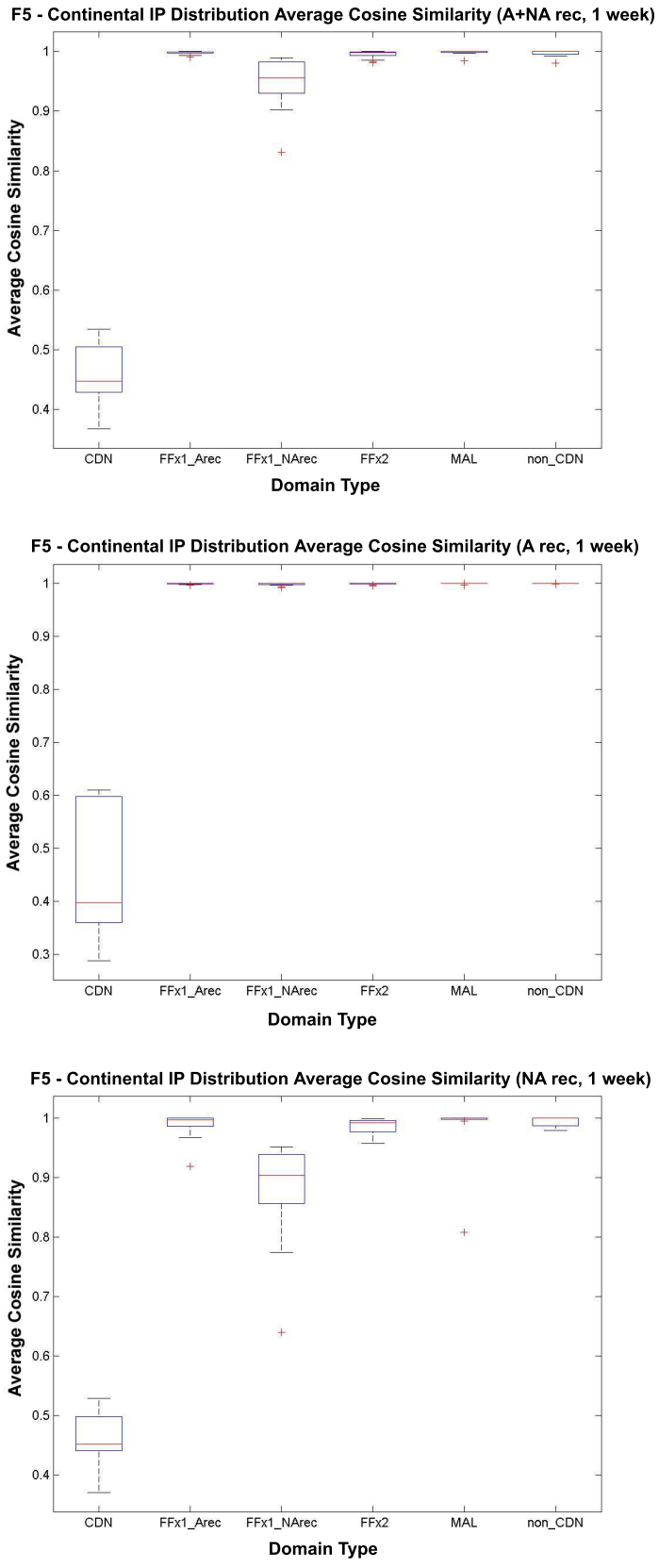


Figure 3.13: Continental IP Distribution Average Cosine Similarity (1 week)

3.3.7 IP Recruiting

In this subsection, we study the distinct strategies employed by FF, CDN, and non-CDN domains when advertising IPs to DNS servers. For a given domain, we assigned a unique IP index to each newly-seen IP in the DNS-query results across all DIGGER nodes. This IP index is plotted against time for example FFx2, CDN, non-CDN, and MAL domains in Figs. 3.14–3.17.[3] The points in the graphs represent when a new IP was returned in a DNS query on a global scale. Therefore, the slope of each curve demonstrates the rate, or speed, with which a domain seems to globally "recruit" more IPs. Notice, when we discuss recruitment, we mean the *apparent* recruitment of IPs based on the DNS-query results, not the actual recruitment of bots via compromising computers.

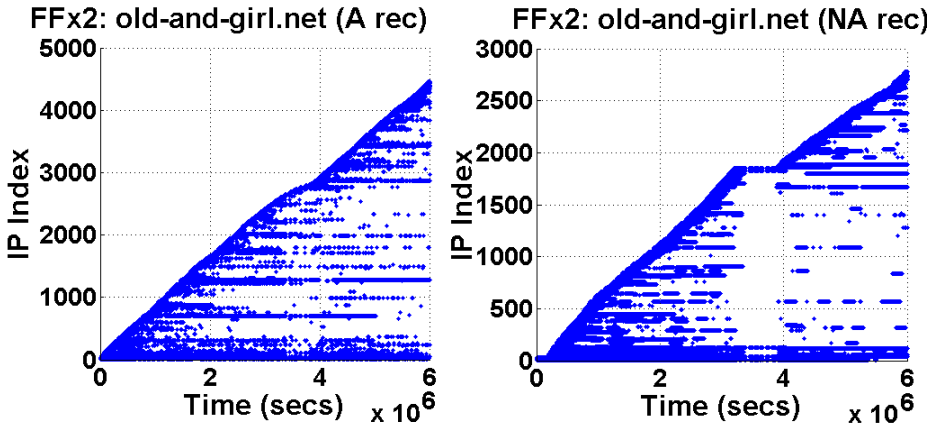


Figure 3.14: Global IP usage for example FFx2 domain

Recruitment Speed refers to the speed (or rate) at which one observes new, unique IPs for a given domain when monitoring that domain's DNS queries over time. Fig. 3.14 shows how a FFx2 domain slowly and nearly continuously accrues unique IPs over its entire online lifetime, with short, intermittent periods of stability. These results indicated that FF domains must constantly advertise new IPs to help ensure reliable delivery of their nefarious content. In addition, the bots used by FF

[3]Plots for FFx1_Arec and FFx1_NArec domains are excluded since they are essentially specific subsets of FFx2 domains.

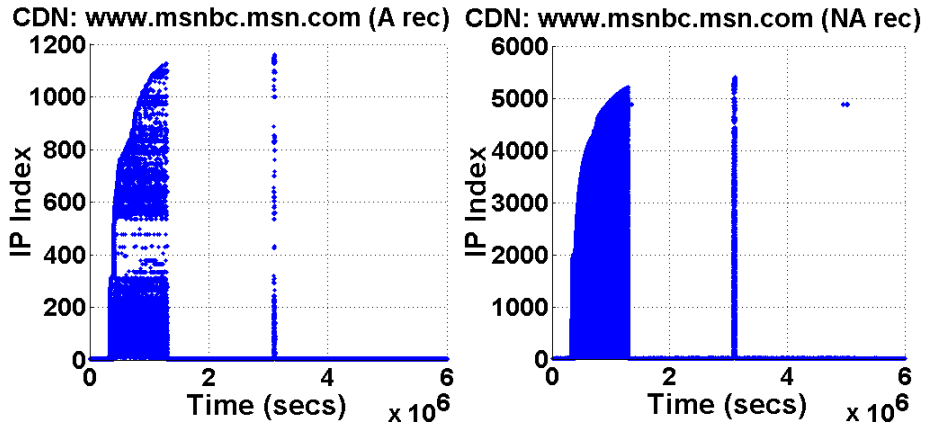


Figure 3.15: Global IP usage for example CDN domain

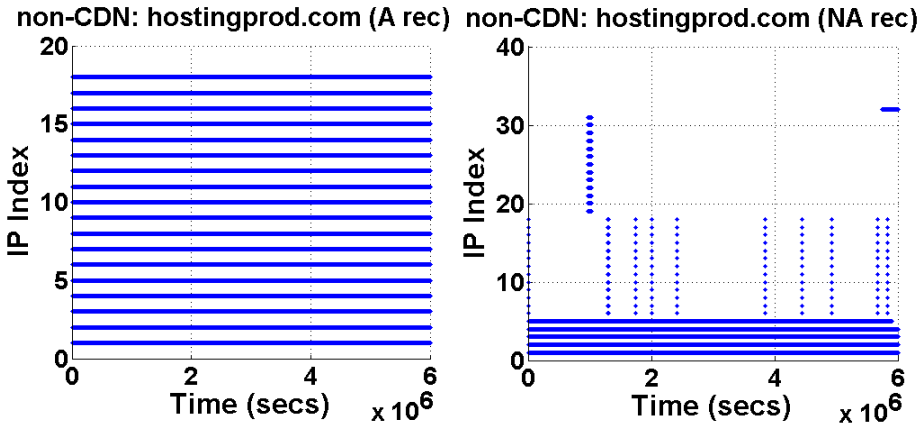


Figure 3.16: Global IP usage for example non-CDN domain

domains may obtain dynamic IP addresses from their Internet Service Provider (ISP). Consequently, a bot may be assigned different IPs over time, causing our DIGGER nodes to observe the apparent recruitment of new IPs; this effect, called *DHCP churn*, is not present for valid domains using stable servers with static IPs.

Meanwhile, when viewed globally, we have discovered that CDN domains (Fig. 3.15) achieve a much faster recruitment speed, indicating that they advertise IPs from a large pool of stable IP addresses, which they rotate quickly and efficiently for performance purposes, such as load balancing. Since CDNs advertise their IPs in a geographically-conscious manner (e.g., a DNS query in Asia will often result in a different set of IPs than a query in Europe), DIGGER's global perspective observes

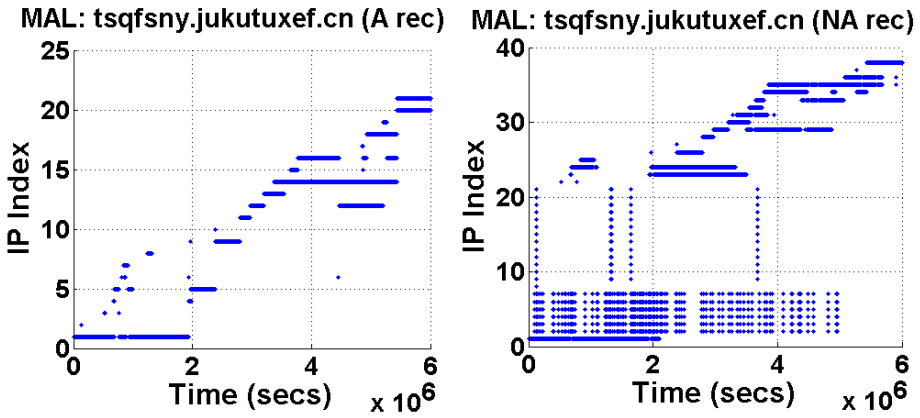


Figure 3.17: Global IP usage for example MAL domain

most of the CDN's IPs in a short period of time. In contrast, FF domains, using necessity-based DNS advertisement, advertise the same pool of IPs irrespective of the DNS servers' geographic locations. Thus, while FF domains may change their advertised IPs as rapidly as CDNs, DIGGER's global perspective doesn't permit it to observe many more IPs than at any given local vantage point, resulting in the comparatively slower IP-recruitment rate.

Non-CDN domains (Fig. 3.16), on the other hand, hardly recruit any additional IPs over time. Rather, their IP pools consist of a small number of stable servers that are almost simultaneously advertised to DNS servers around the world.

MAL domain (Fig. 3.17) often demonstrate a slow and somewhat steady recruitment of IPs. This behavior is likely the result of the MAL domains' malicious activities being detected and their IPs blocked, requiring them to register fresh IPs with DNS to maintain content availability. Unlike FF domains, which recruit thousands of IPs, MAL domains recruit only tens of IPs over ≈3.5 months. This drastic difference should prove beneficial in distinguishing MAL domains from CDN and FF domains.

To determine if a domain's recruitment speed can serve as a useful differentiating feature, let us quantify it as follows:

F6 First, we calculate the domain's recruitment time, denoted as T_r. We consider a *recruitment point* to be a point in time where we have observed a *new* IP address (i.e., one that hasn't occurred earlier in time). If the difference in time between two consecutive *recruitment points* is less than the threshold, we add the it to T_r. Let N = the total number of unique IPs observed globally for a domain. Then, the IP-recruitment speed (F6) is calculated as:

$$\text{F6} = \frac{N}{T_r} \tag{3.7}$$

In those instances where all of a domain's IPs are observed instantaneously, resulting in a $T_r = 0$, we set F6 to 0, indicating that the domain has no recruitment time and thus no recruitment speed.

Using Eq. (3.7), we calculate F6 over a representative week for the 90 manually identified domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.18. From the figure, we find that, when viewed over only a single week, feature F6 is not as useful a differentiating feature as originally indicated. It seems a better representation of domains' recruitment behavior is required to capture the various recruitment strategies employed by the different domain types. In an effort to find such a representation, we next explore the IP-recruitment period.

Recruitment Period represents the amount of time during which new IPs are witnessed for a given domain when monitoring that domain's DNS queries over time. Our data indicates that non-CDN domains (Fig. 3.16) have almost no recruitment period; a small pool of very stable IPs are advertised initially and used throughout the lifetime of the domain. On the other hand, the fast recruitment speed of CDN domains causes DIGGER to quickly observe most of their available IPs, resulting in a short recruitment period at the onset of monitoring followed by a longer, stable period consisting mainly of previously seen IPs. From Fig. 3.15, we notice that the

CDN's recruitment period is smaller than its total online period. After its initial recruitment period, the CDN domain stabilizes and advertises a much smaller set of IPs before a quick advertisement spike followed by another stable period; the stable period looks like a gap in the graph, but closer examination reveals a small set of IPs with low IP indices (i.e., the earliest observed IPs). We have also discovered, as shown in Fig. 3.14, that the fluxy records for FF domains recruit new IPs for nearly the entire duration of the domains' online period, with only short, intermittent periods of stability. This constant IP recruitment is a result of the unreliable nature of the compromised computers serving as bots.

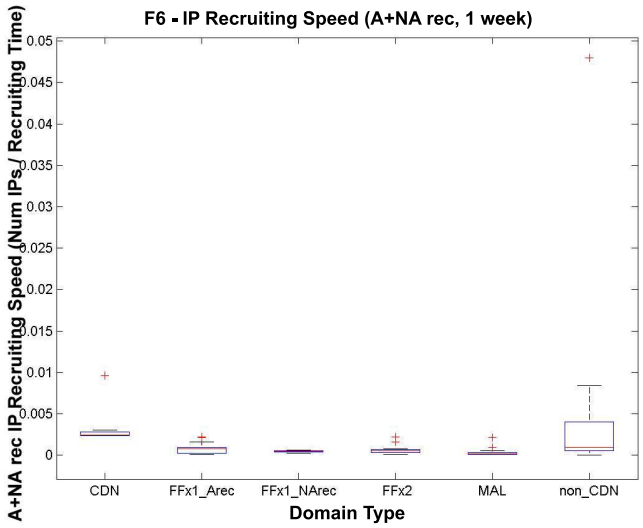
In order to better evaluate a domain's recruitment period as a distinguishing feature, let us quantify it as follows:

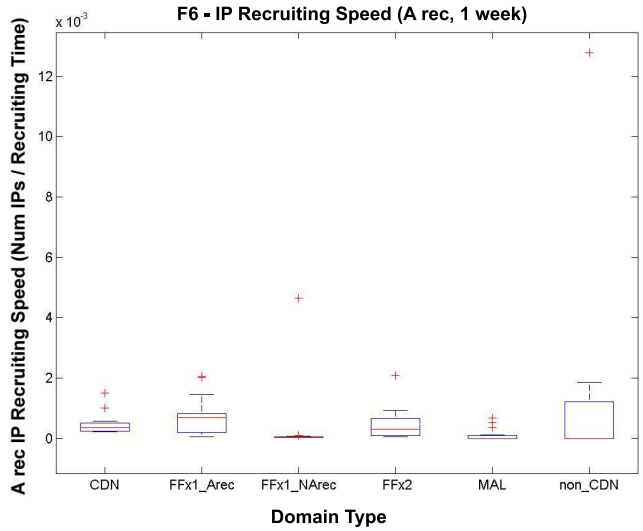
F7 First, we calculate a domain's online time, denoted as T_o, as the amount of time we consider the domain to be online. Analyzing all available DNS-query data from all nodes, we consider an *online point* to be a point in time where we have observed IP addresses. If the difference in time between two consecutive *online points* is less than a threshold of several hours, we add it to the T_o. Then, using T_r as in Eq. (3.7), the recruitment period (F7) is calculated as:

$$\text{F7} = \frac{T_r}{T_o}. \tag{3.8}$$

Using Eq. (3.8), we calculate F7 over a representative week for the 90 manually identified domains' A-, NA-, and A+NA-record IPs, shown in Fig. 3.19. From the figure, we find that F7 achieves much better success as a distinguishing feature than F6. For the A-rec IPs, we find that FFx1_Arec and FFx2 domains recruit new IPs for a greater percentage of their total online time than the other domain types. In particular, almost all of the FFx1_Arec domains tend to recruit A-rec IPs for over 90% of their total online time. While there is a greater variation in the FFx2 A-rec

IPs, we can still see that 50% of them recruit IPs nearly constantly (i.e., for almost 100% of their online time). Of the remaining 50%, most of them recruit A-rec IPs for over 80% of their online time. On the contrary, non-CDN and MAL domains almost never recruit new A-rec IPs when analyzed for a single week. FFx1_NArec domains, which behave mostly like non-CDN domains in the A rec, have a much lower recruitment period in their A-rec IPs than the FFx1_Arec and FFx2 domains. However, they seem to have a higher recruitment period than the non-CDN and MAL domains. This is likely due to their limited use of IP overlap, using a few select A-rec bots as NA-rec IPs. Finally, CDN domains have a smaller recruitment period in their A rec than the FFx1_Arec and FFx2 domains, but a significantly larger one than the other domain types. The IP-recruitment period of CDNs in the NA rec is even more pronounced, typically achieving results greater than 80%—similar to the FFx1_NArec and FFx2 domains. As expected, we find that FFx1_NArec domains demonstrate a high recruitment period (typically greater than 90%) in their NA rec, while FFx1_Arec domains have below 10% in most cases. As with the A-rec IPs, FFx2 domains demonstrate more variance in their NA rec IP-recruitment period; 50% of them easily recruit IPs for more than 95% of their online time, while the other 50% demonstrate recruitment periods of 20% or more. Conversely, in the NA rec, FFx1_Arec domains, non-CDN domains, and MAL domains often recruit NA-rec IPs for less than 20% of their total online time. When calculating the recruitment period on the A+NA rec (i.e., the combined A-rec and NA-rec IP pool), we find that CDN and FF domains, with recruitment periods typically greater than 75%, are easily distinguishable from the non-CDN and MAL domains, whose periods generally fall below 40%. Unlike recruitment speed (F6), this should prove a powerful metric for identifying non-CDN and MAL domains from the other domain types.





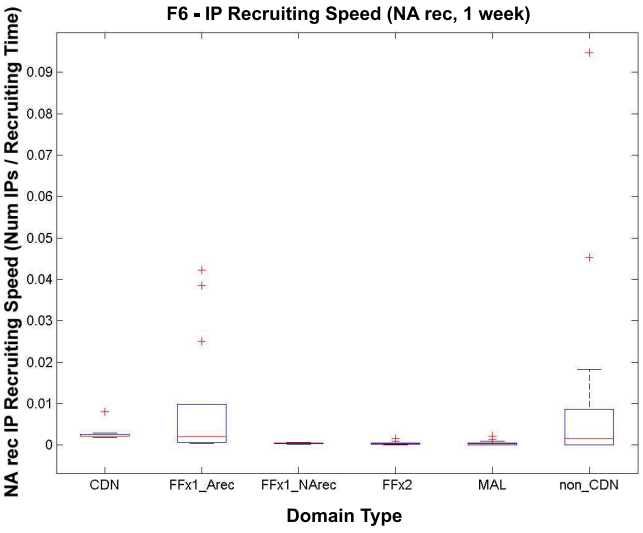
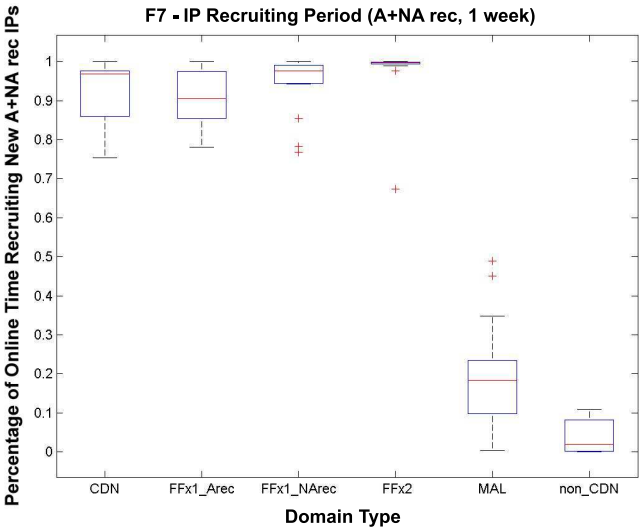
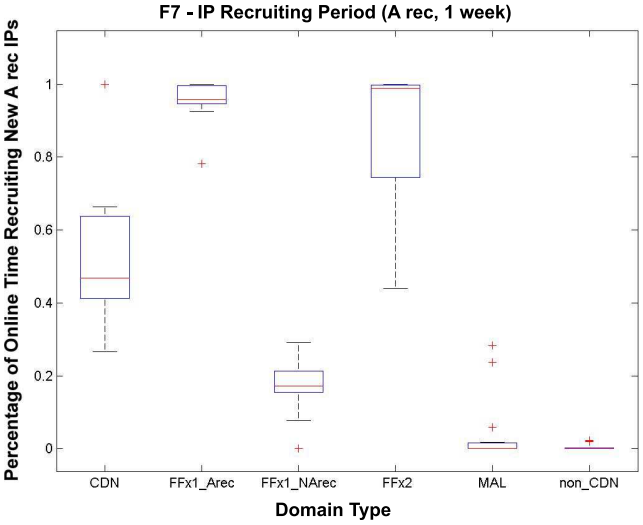


Figure 3.18: IP-Recruitment Speed (1 week)





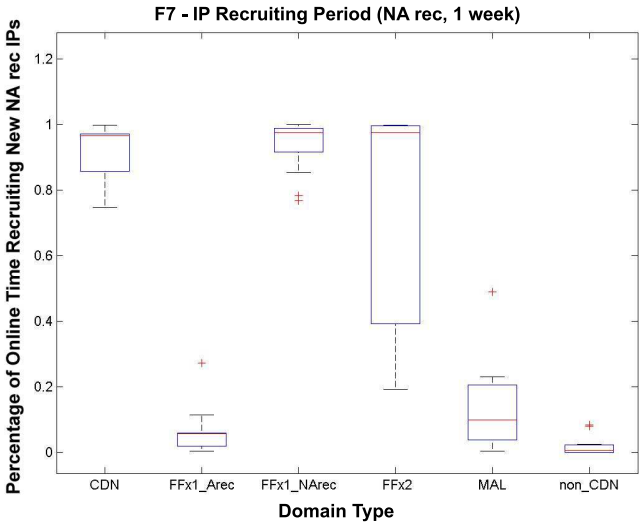
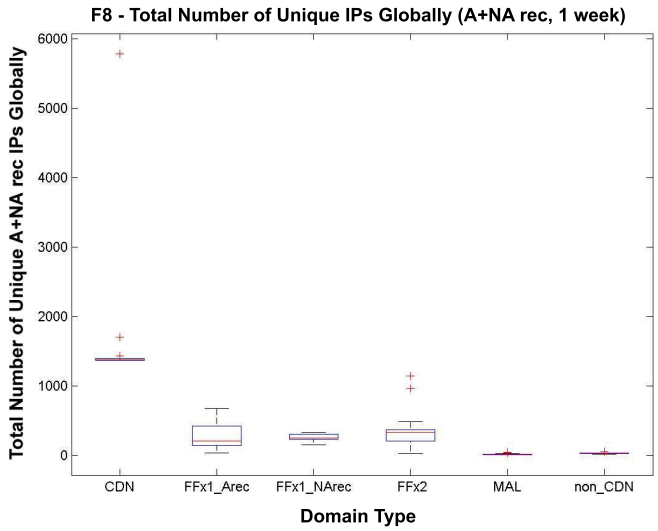


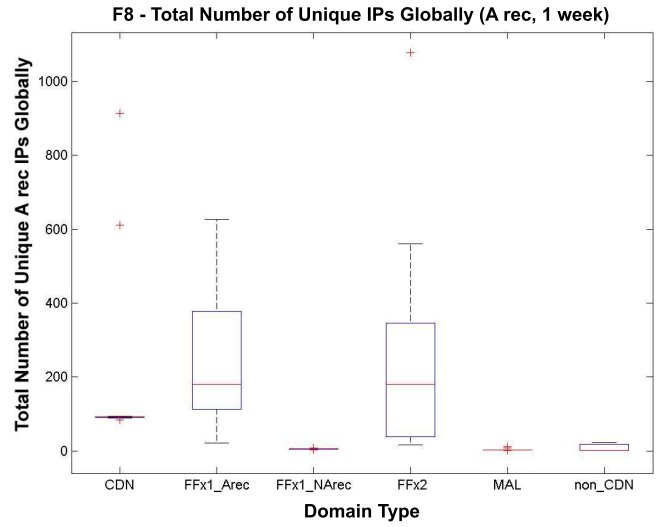
Figure 3.19: IP-Recruitment Period (1 week)

3.3.8 Total Number of Unique IPs Globally

The total number of unique IPs seen across all DIGGER nodes (i.e., feature F8) proves remarkably apt as a metric for distinguishing non-CDN and MAL domains from CDN and FF domains. This is because, compared to CDN and FF domains, non-CDN and MAL domains advertise only a few stable content and name servers with DNS. Since non-CDN and MAL domains' A-rec and NA-rec IPs are seen ubiquitously around the globe, their total number of unique IPs observed by the DIGGER nodes over time will be meager. Table 3.3, which shows the number of IPs in the A and NA recs for examples of the different domain types, demonstrates this effect. The CDN and FFx2 domains display abundant IPs in their A and NA recs. While the FFx1_Arec domains possess a modest number of NA-rec IPs, they have a substantial number of A-rec IPs—a clear distinction from non-CDN domains. The opposite holds true for the FFx1_NArec domain; the small number of IPs cause its A rec to resemble a non-CDN domain, while the much larger number of NA-rec IPs betrays this guise.

We plot F8 over a typical week in Fig. 3.20 for the 90 manually identified domains' A-, NA-, and A+NA-record IPs. From the figure, we can clearly see that the FF and CDN domains utilize more unique IPs globally than the non-CDN and MAL domains. While the FF domains tend to use more IPs on average in their A recs, this honor belongs to the CDN domains for the NA recs. It appears that the combined A+NA recs can be used to distinguish CDN and FF domains from the other domain types, indicating that this could serve as a useful feature in identification.





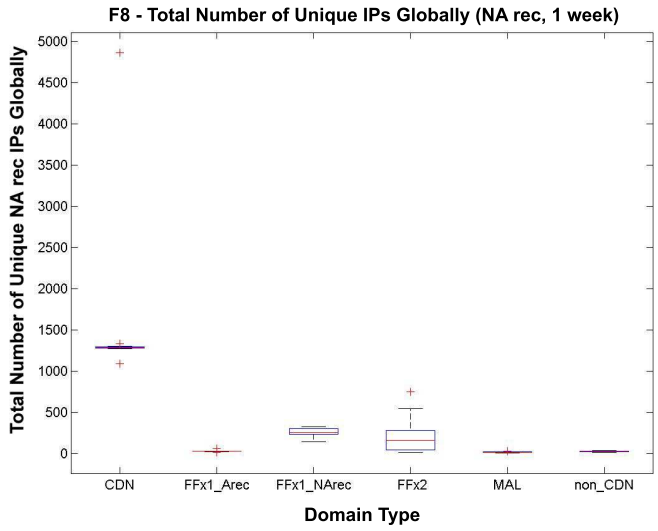


Figure 3.20: Total number of unique IPs globally (1 week)

3.3.9 Reverse DNS lookup and TTL

The last two features we will discuss seem to be obvious candidates for use in classification: the reverse DNS lookup result (i.e., the reverse DNS name) and the TTL values of the A and NA recs. Clearly, if the reverse DNS lookup on a domain contains suspicious words typical to home computers (e.g., comcast, dynamic, dial-up, etc.), it is a strong indicator that the IP belongs to compromised computer, or bot. Because an IP's reverse DNS name is set by the IP's ISP and not the owner of the domain, it cannot be faked by a botmaster. This makes it a fairly useful metric for identifying bots. Unfortunately, the reverse DNS lookup is highly unreliable. Often, a reverse DNS lookup will not return a result, thus providing no insight into the actual identity of the suspect IP. Additionally, we don't have a complete list of suspicious words, and occasionally, the presence of such words may not be indicative of a bot; often, it is only after thoroughly analyzing the DNS data in conjunction with the reverse DNS words that we can determine them to be bad, strengthening a malicious classification. Therefore, we have decided not to incorporate the reverse DNS name for automatic domain classification. Instead, when present, we use it to help reinforce or confirm our manual identification of the different domain types. By omitting it from our automatic identification, we hope to gain a better insight into the potential of the more reliable classification features.

The A and NA recs' TTL values also appear highly useful for differentiating between the domain types. CDNs and FF domains tend to use small TTL values, affording them a high level of control over the domain's IPs. CDN domains use this extra control for load balancing and reliable content delivery. FF domains are really only concerned with reliable content delivery in the presence of unreliable content servers (i.e., bots). Non-CDNs, unperturbed by these concerns, use much longer TTL values for their stable content and name servers. However, unlike many of the other features we have previously explored, the TTL value is not an uncontrollable

consequence of a botnet. While it is difficult for a botmaster to mimic features such as a CDN's location-aware DNS-advertisement strategy or a valid domain's recruitment speed/period without sacrificing content availability, this is not the case with the TTL value. An IP's TTL value is set by the owner of the domain. A botmaster can easily increase the average TTL value for its A or NA records without sacrificing the availability of the malicious content. By setting a short TTL value for some IPs and very large TTL values for others, the average TTL of FF domains can be made to resemble the average TTL of non-CDN domains without sacrificing the fine level of control over some of the IPs. Those IPs with large TTLs (used to inflate the average value) could belong to more reliable bots; they could just as easily be bogus IPs that don't resolve to anything. So long as some of the IPs (presumably those with the shorter TTLs) resolve to online bots, the malicious content can still be reached. While we could try more complicated methods of measuring the TTL values to account for this inflation technique, it would be just as easy for botmasters to come up with another clever way to circumvent our metric. Botmasters simply have too much control over the TTL value for it to be a reliable feature for classification. Therefore, we have decided not to use it as such. It should be noted that other features, like the recruitment speed and period, cannot be as easily manipulated by the botmaster since the unstable bot IPs necessitate constant recruitment.

3.4 Analysis of the Different Domain Types

In this section, we will first describe the multi-level, multi-week, linear SVM classifier we use to identify the different domain types present in DIGGER's ≈3.5 months of global DNS data. The SVM classifier allows us to quickly identify the various domain types out of the 5,171 monitored domains. Next, we will compare the different domain types in terms of the previously discussed features and in their overall breakdown of the monitored domains.

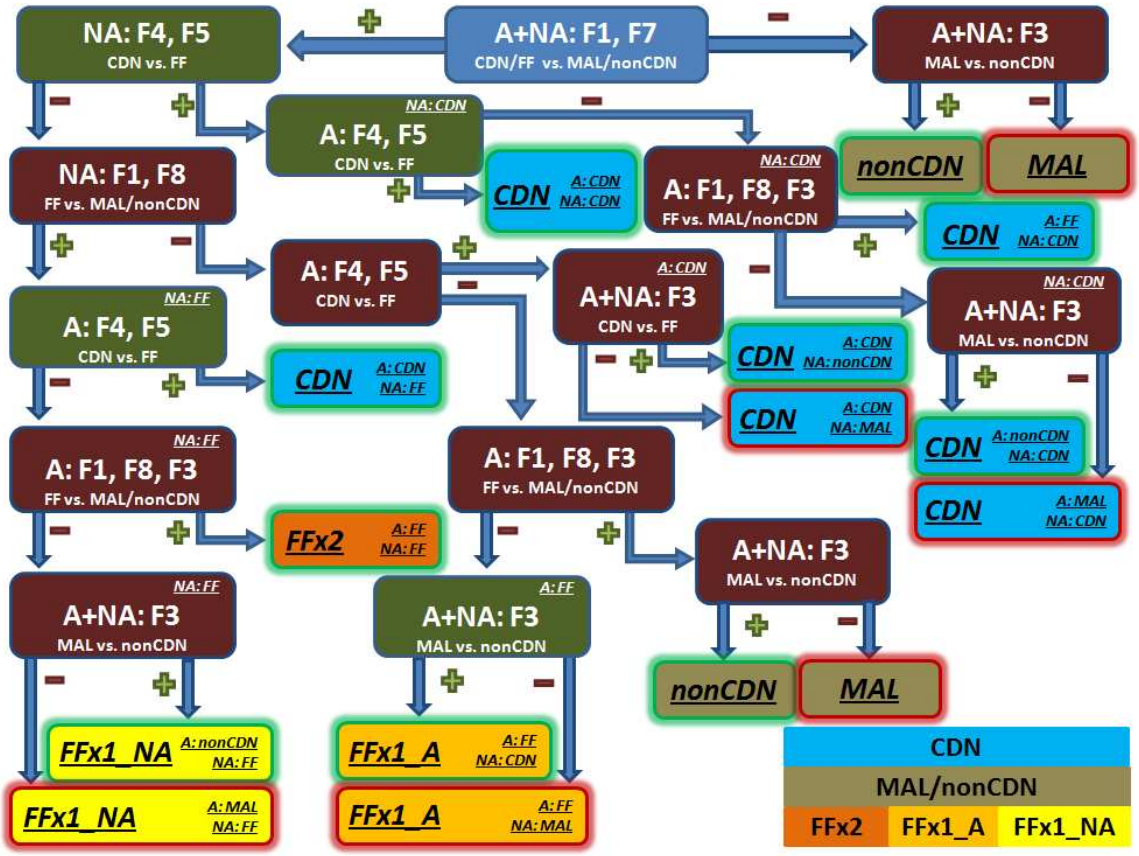


Figure 3.21: Flow chart for multi-level, linear SVM classifier

3.4.1 SVM Classifier

The purpose of our classifier is not to serve as a detector. The purpose of our classifier is for the classification of a large number of domains, allowing us to better analyze and understand their various DNS-advertisement strategies and uncover trends in FF domains' DNS behavior. While DIGGER's global vantage point of 240 nodes spanning 4 continents can provide us with an unprecedented view of DNS-advertising strategies, requiring such a massive distributed system for detection is impractical. Likewise, our monitoring period of ≈3.5 months is far too long to prove useful for detection; by the time the FF domains are detected, they would have had months to perpetrate their scams. However, by analyzing these domains from so many different vantage points for such an extended period of time, we can develop

SVM Node		SVM Equation
A+NA:	F1, F7	*-5.59 + 4.46(F7_A+NA) + 0.16(F1_A+NA)*
	F3	*-1(F3)*
A:	F4, F5	*3.40 + -3.44(F5_A) + -2.40(F4_A)*
	F1, F8, F3	*-1.41 + -0.36(F1_A) + 0.34(F8_A) + -0.19(F3)*
NA:	F4, F5	*3.29 + -3.50(F5_NA) + -2.18(F4_NA)*
	F1, F8	*-5.29 + 0.16(F8_NA) + -0.11(F1_NA)*

Table 3.4: Linear SVM Equations for multi-level classifier

an off-line classification system capable of quickly analyzing the massive amount of data and accurately identifying the different domain types.

3.4.1.1 Challenges

In designing our classifier, there were a couple of challenges we had to overcome. First, despite DIGGER gathering global DNS data for ≈3.5 months, not all domains we monitor are actually active for that entire duration. In particular, FF and MAL domains, which are eventually detected by other means, are blocked and unexpectedly go offline (i.e., their DNS results no longer contain valid IP data). As a result, comparing features extracted over the *entire* monitoring period could give an inconsistent and incorrect representation. For example, benign domains, such as CDNs and non-CDNs, will have many more data points than many malicious MAL and FF domains. For accurate comparisons, the different domain types' features must be extracted over a normalized time period when the domains are active and online. Second, despite discovering differentiating features, there can be situations when benign CDNs or non-CDNs temporarily appear like FF domains and vice versa. This can be an artifact of CDN domains utilizing load-balancing over location-aware strategies in their DNS advertisement, causing features F4 and F5, which encapsulate this location-aware behavior, to appear similar to the other domain types. Additionally, FF domains that have been detected and gone offline can have their domains blocked

by their ISPs or DNS. Visiting these domains results in a web page informing unsuspecting users that the malicious site has been shutdown for their protection. Such warning pages are typically hosted on traditional servers, causing their DNS behavior to resemble non-CDN domains. Clearly, these two challenges are not entirely independent; when making a classification decision, the greater the duration of DNS activity examined, the more likely it will include and be influenced by some of these irregular DNS-advertisement strategies.

In designing our classifier to overcome these challenges, we make use of the following insights. First, DIGGER's global vantage point and extensive monitoring period has provided us with a plethora of DNS-advertisement data. As such, even when FF domains go offline, DIGGER will have already accrued *some* DNS data from when the domain was active, albeit, not the full ≈3.5-months worth of data. Consequently, by using a smaller, consistent examination period, we can ensure that the extracted behavioral features used for comparison occur while the domains are still active and online. Second, with such a comprehensive dataset, even domains periodically demonstrating irregular DNS-advertisement behavior will only do so for a portion of the entire monitoring period. By extracting and examining features from those periods when the domains' are *not* expressing deviant behavior, we can better compare and contrast their various DNS advertisement strategies.

Based on these insights, we have designed our classifier to utilize a 1-week examination window when extracting the behavioral features discussed in Sections 3.3.3—3.3.8. We chose a 1-week window for two reasons. First, it allows us to extract multiple weeks for examination from the entire monitoring period, increasing the probability that any unorthodox behavior will be confined to a particular week and not influence the features extracted from other weeks. Second, a week-long duration is sufficient for the FF domains—especially the less fluxy domains—to differentiate themselves from the benign domains. Unfortunately, since MAL domains are es-

sentially non-CDN domains that recruit new IPs once or twice a week when they are ultimately detected and blocked, the week-long examination window will prove inadequate for distinguishing them from non-CDN domains. However, since MAL domains' recruiting behavior is a direct consequence of being detected and blocked by other systems, and since the focus of this paper is to better understand the behavioral features distinguishing FF domains from other domain types, this is acceptable. While we can use feature F3 (IP Overlap) to determine which non-CDN domains are definitely MAL, some non-CDN domains identified by our classifier could actually be MAL domains not utilizing IP overlap. Consequently, for the remainder of this section, non-CDN domains should be considered the combined domain type non-CDN/MAL.

3.4.1.2 SVM Design and Implementation

As previously mentioned, our classifier examines all available DIGGER data for domains by extracting features over week-long periods when the domains' are active. Consequently, this results in most domains containing multiple weeks of feature data for which we can attempt to arrive at a classification decision. Choosing representative weeks for 15 manually identified domains of each type, we built a training dataset of 90 domains. The 10- and 5-fold cross validation scores were used in conjunction with the trained SVM equations' weights to arrive at an optimal ordering for domain classification and determine which features to use at each stage of the classifier. This was accomplished by testing all possible feature combinations and orderings for classifying domains. The resulting multi-level SVM classifier's decision tree is shown in Fig. 3.21, and Table 3.4 shows the weights for the individual linear SVM equations used at each decision node.

When classifying a domain for subsequent analysis, we first apply the classifier to each individual week of DIGGER data for which the domain was active and online.

Next, based on each domain's weekly classification results, we arrive at an overall classification for the domain. As mentioned previously, domains can sometimes exhibit DNS behavior uncharacteristic of their normal advertisement strategies. For example, CDNs can favor load-balancing over location-aware strategies or FF domains can become blocked and appear more like non-CDN/MAL domains. In addition, the further classification of FF domains into their various advertisement strategies (i.e., FFx2 vs. FFx1_Arec vs. FFx1_NArec) is determined by which record type (A or NA) exhibits fluxy behavior; this can be complicated by the fact that sometimes FF domains temporarily alter this strategy or reduce fluxiness for a particular record type. When viewed in discrete, weekly segments, this can result in typically FFx2 domains intermittently exhibiting FFx1 behavior and being classified as such. To account for these irregular weekly classification results and arrive at a more accurate and representative classification for the domains, we make use of several insights.

First, the location-aware DNS-advertisement strategy espoused by CDN domains is inherently difficult—if not impossible—for FF domains to successfully duplicate. Relying only on unreliable, compromised computers, FF botnets are at the whim of their constituent bots' online availability. Even if a large botnet comprises enough bots sufficiently dispersed around the world to theoretically duplicate such a strategy, the bots' intermittent connectivity makes actually achieving this feat exceptionally difficult. Not all compromised computers possess 24-hr online connectivity, and for those that do, diurnal usage patterns can limit their reachability as they are shutdown or put to sleep/hibernation when not in use (e.g., when people are sleeping for the evening). Consequently, while CDN domains that temporarily favor load-balancing over location-aware advertisement strategies may mistakenly be classified as FF domains, the converse is highly improbable and has never been witnessed in our experience. Thus, any domain which demonstrates sufficient location-aware behavior (i.e., features F4 and F5) to be classified as a CDN domain during one of the

examined weeks is considered a CDN domain; any other weekly classification results are trumped in favor of this uniquely distinguishing and nearly impossible to replicate DNS-advertisement strategy. Second, since FF domains, once blocked, can appear as non-CDN/MAL domains, a weekly classification of FF is favored over competing non-CDN/MAL classifications. This allows us to identify the domain for what it truly is: a previously active FF domain that has since been detected and blocked. Third, for competing FF classifications, we perform a majority vote to determine which type of FF advertising strategy is predominantly used. In the case of a tie, we examine the magnitudes of the classifier's many SVM output branches, choosing the most confident FF classification decision. Lastly, recall that MAL domains are essentially non-CDN domains that recruit new IPs when their nefarious actions have been detected and their IPs blocked. As a result, they demonstrate malicious IP-recruitment behavior infrequently, recruiting only a handful of IPs over the entire monitoring period. Since our classifier extracts features based on a week of DNS data at a time, the malicious IP-recruitment behavior of MAL domains is seldom demonstrated. Thus, non-CDN and MAL domains often appear identical when viewed over a week, with the only differentiating feature being the presence of IP overlap (F3). Therefore, our classifier makes use of this simple observation to identify domains with IP overlap as definitely MAL. However, since not all MAL domains demonstrate IP overlap, our classifier is able to partition the non-CDN and MAL domain types into two sets: 1) those that demonstrate IP overlap and, therefore, are certainly MAL domains and 2) those that do not demonstrate IP overlap and could be either non-CDN or MAL domains. Since our focus is on the understanding of FF DNS-advertising strategies and MAL domains are essentially non-CDN domains used for nefarious purposes, this lack of precision in distinguishing benign non-CDN domains from their malicious counterparts (i.e., MAL domains) is acceptable.

By training our classifier and aggregating its weekly results in this manner, our

multi-level, multi-week classifier is more robust to irregular behavior and blocked domains. Having arrived at a confident classification decision, we once again make use of the SVM branches' output magnitudes to determine the best representative week for each domain. Next, we present a comparison of these representative weeks for the the different domain types in terms of their behavioral features. We also examine the distribution of the various domain types relative to our set of monitored domains, exposing trends in DNS-advertising strategies.

3.4.2 Results

We applied our classifier successfully to the entire dataset and identified the various domain types. First, we will examine the relative distribution of the domain types within our dataset. Next, choosing the most representative week for each domain, we will compare the features previously discussed to determine how well they differentiate in the context of this large population of domains.

3.4.2.1 Domain Type Distribution

The relative distribution of the various domain types is shown in Table 3.5. Because the domains monitored by DIGGER come primarily from spam emails and online repositories of malicious domains, it is not surprising that the primary domain type in our dataset is non-CDN/MAL domains, with the majority of them being decidingly MAL due to IP overlap. The use of traditional content and name servers for delivering malicious web pages is still the most popular and dominate strategy, accounting for over 95% of the domains monitored by DIGGER. This is presumably due to their ease of use; bad guys can easily find less-than-reputable hosting services in a variety of countries, permitting simple deployment of a malicious website. When hosting services are pressured to shut down the malicious domain, it's arbitrary for the malicious content to be moved to another hosting service to continue its nefarious

scams. Likewise, the small amount of legitimate, benign CDN domains is a consequence of DIGGER's domain sources of being ill repute. Certainly, some of the spam emails could contain legitimate web pages hosted on CDNs, however, we discovered that this occurs infrequently.

The additional level of misdirection and the nearly limitless supply of bot IPs make FF domains appealing for nefarious purposes, despite their more diligent maintenance requirements. Thus far, it has been primarily FFx1_Arec domains observed in the wild, and their popularity is supported with our findings: ≈40% of the FF domains are FFx1_Arec. FFx1_Arec domains are likely the most popular since they provide botmasters the greatest return on their investment, affording them an additional layer of misdirection without the hassle of maintaining volatile botnet NSes. Botmaster must still monitor the domain and replace the botnet IPs to avoid an interruption of service, but this task is greatly simplified with the use of stable NSes. Unfortunately for botmasters, security professionals have become aware of the FFx1_Arec botnet technique, devising clever detection strategies. While the botnet provides a steady source of fresh A-rec IPs, the NSes can still be blocked, crippling the botmaster's control until new NSes can be acquired.

One means of overcoming this limitation is to adopt a FFx2 advertising strategy, using bots in the domain's NA rec as well. This technique seems to be gaining support among botmasters, with FFx2 domains composing ≈27% of the FF domains observed. FFx2 domains improve upon FFx1_Arec domains by providing an additional layer of misdirection, further protecting the botmaster. Clearly, FFx2 domains require a more diligent management effort than FFx1_Arec domains; in addition to the A rec, the botmaster must constantly replace IPs in the NA rec as well. However, this extra effort also makes FFx2 domains more difficult to subvert, protecting the NSes against simple countermeasures such as IP blocking. Interestingly, when we analyzed the identified FFx2 domains, we discovered a spectrum in the amount of NA-rec

fluxiness botmasters were incorporating. Obviously, there were domains that were incredibly fluxy in both record types, as demonstrated by *old-and-girl.com* (Fig. 3.14). Such FFx2 behavior is essentially what we had envisioned when applying the better-known, fluxy A-rec behavior to the NA rec. While it's interesting to observe these aggressive FFx2 domains in the wild, it was the FFx2 domains at the other end of the spectrum that proved more insightful. As an example, we present the more modest FFx2 domain *ehuytyt.cn*, shown in Fig. 3.22. With over 2,500 unique A-rec IPs, *ehuytyt.cn* is considerably more more fluxy in its A rec than its NA rec. Using stable bot IPs from its A rec for roughly a quarter of its NA-rec IPs, FFx2 domains like *ehuytyt.cn* benefit from the increased control and stability provided by traditional NSes, while simultaneously enhancing the domain's resilience to subversion—for a minimal increase in management—through the use of botnets.

Another interesting discovery is the apparent popularity of FFx1_NArec domains, accounting for ≈33% of the total FF domains observed. Surprisingly, this is a larger share than the FFx2 domains. It seems that botmasters have become aware of security professionals analyzing domains' A recs for FF behavior. Consequently, they have migrated the fluxy behavior to the NA recs, where it is less likely be noticed. Fig. 3.23 is a typical example of the FFx1_NArec domains identified by our classifier. It demonstrates a MAL domain strategy for its A-rec IPs and a FF strategy for its NA-rec IPs. This results in the domain appearing more benign when its A recs are analyzed, while providing the botmaster with a fine level of control over the NSes. Should the domain's malicious activity be detected and the A-rec IPs blocked, the botmaster, having retained control over the NSes, can easily replace the IP's with minimal service interruption. The implication of this discovered behavior is straightforward: both record types must be monitored for fluxy behavior in order to quickly identify FF domains and their botnets. A real-time monitor analyzing only domains' A recs will not identify FFx1_NArec domains as fluxy, and it could take days for the

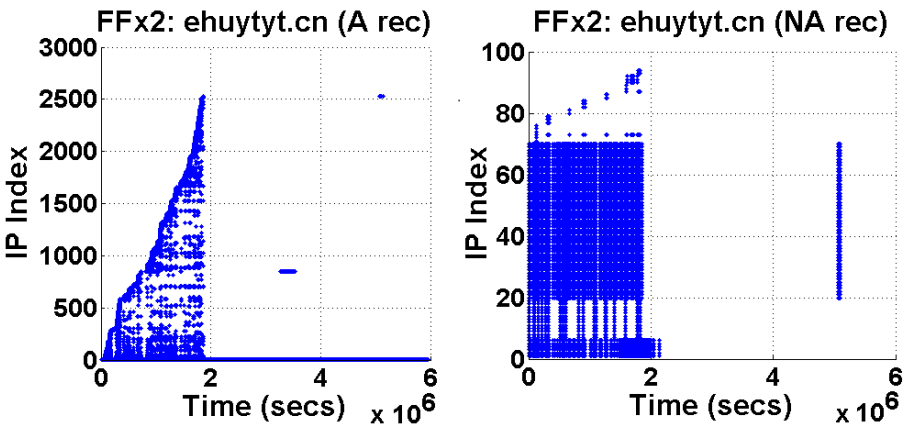


Figure 3.22: Classified FFx2 domain

A rec's MAL domain behavior to display its slow and steady IP recruitment; even then, the observed recruitment is a side effect of others detecting the malicious domain and blocking its IPs. However, a real-time detection system monitoring NA recs for fluxy behavior could identify the domain to be FF more rapidly—quite possibly before any MAL domain behavior becomes apparent in the A rec. Obviously, the faster malicious domains can be identified, the sooner they can be shutdown or have their nefarious influence mitigated.

Domain Type	Num of Domains	% of ALL	% of FF	% of FFx1
ALL	5,171			
CDN	18	0.35%		
non-CDN/MAL	4,958	95.88%		
MAL	4,341	83.95%		
FF	195	3.77%		
FFx2	53	1.02%	27.18%	
FFx1	142	2.75%	72.82%	
FFx1_Arec	78	1.51%	40.00%	54.93%
FFx1_Narec	64	1.24%	32.82%	45.07%

Table 3.5: Relative distributions of the various domain types

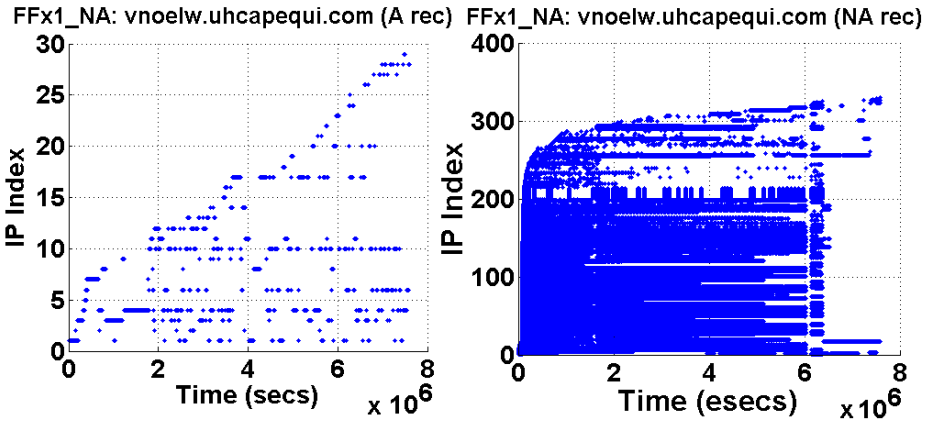


Figure 3.23: Classified FFx1_NArec domain

3.4.2.2 F1 - Average Number of Unique IPs per Node

In Fig. 3.24, we have plotted the CDF for feature F1 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. From the figure, we find that F1 seems to do a decent job at differentiating FFx1_Arec and FFx2 domains from the other domain types in the A rec. However, there are still a decent portion of CDN domains which share similar behavior, complicating classification. This behavioral affinity in F1 between FF and CDN domains is further demonstrated in the NA rec, where CDN domains actually have a greater average number of IPs per node than the FF domain types. In addition, while FFx1_NArec domains certainly possess a greater F1 on average than non-CDN and MAL domains, the distinction is not as strong as would be desired for confident classification purposes. The most promising results come from the combined A+NA rec, where there is a strong line of demarcation separating the FFx1_Arec, FFx2 and CDN domains from the other domain types. From the figure, we observe that in the A+NA rec, over 90% of FFx1_Arec, FFx2 and CDN domains have an average of more than 20 IPs per node; on the contrary, only 20% of FFx1_NArec domains and less than 1% of non-CDN and MAL domains have an average of more 20 IPs per node. These findings confirm that F1 can serve as a powerful feature for differentiating FFx1_Arec, FFx2 and CDN

domains—and to a lesser extent FFx1_NArec domains—from non-CDN and MAL domains.

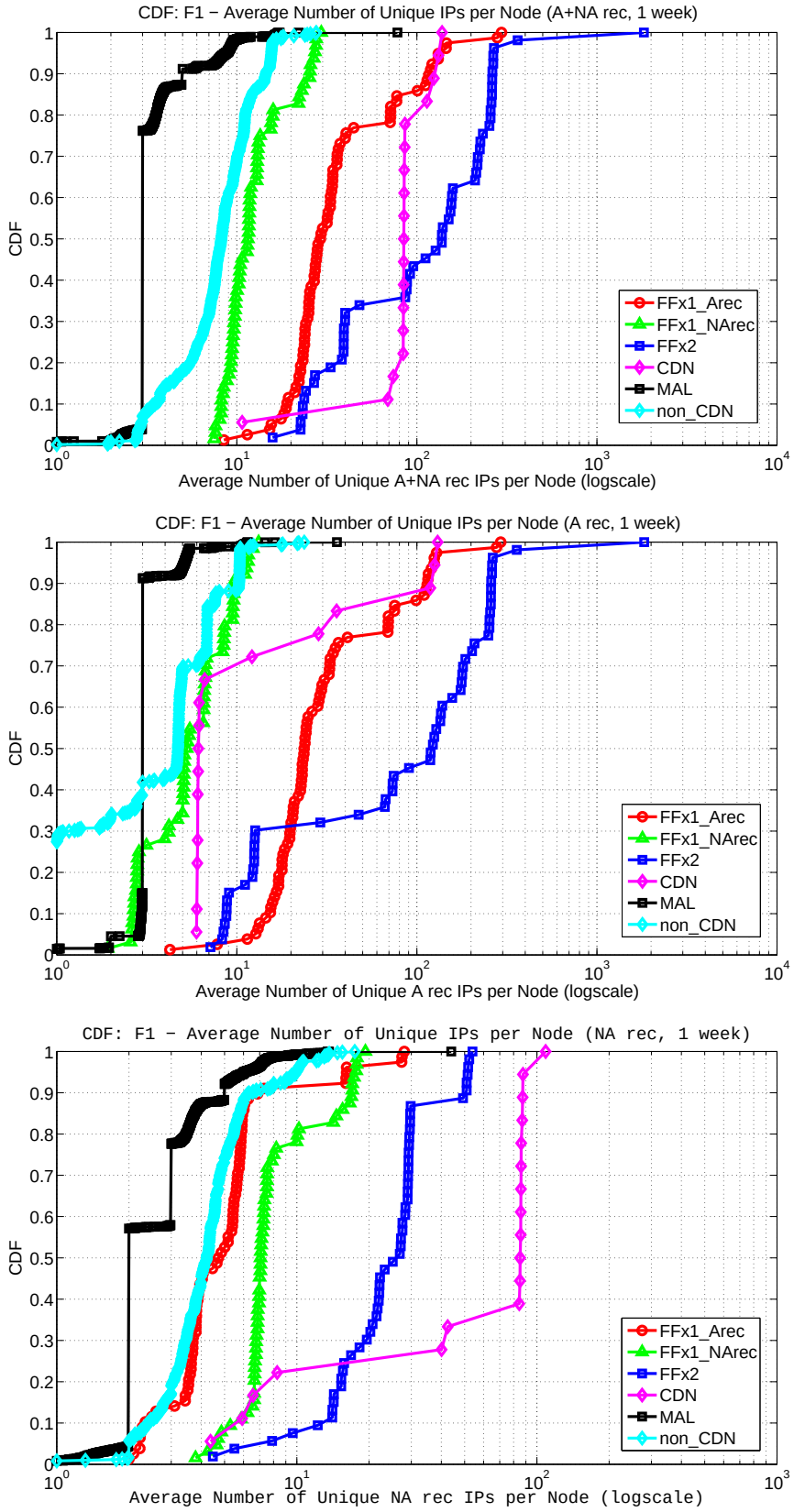


Figure 3.24: CDF of F1 - Average number of unique IPs per node (1 week, log scale)

3.4.2.3 F2 - Average Number of Nodes per IP

In Fig. 3.25, we have plotted the CDF for feature F2 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. As previously anticipated, feature F2 does not appear to be a powerful feature for use in detection. Only in the A rec, where CDN domains' location-aware advertising strategies dominate, does it seem to possess any merit as a differentiating feature. In the A rec, the only domain type that seems to have any comparable F2 values is the MAL domain type. Deeper analysis into these MAL domains with very low F2 values has shown that they are the result of already detected and offline MAL domains. In such a cases, most DNS servers around the world will report empty DNS results, as the domains are no longer advertising with DNS. However, a few, either due to misconfiguration or an effort to protect users by returning a warning page, will return a result. In such cases, the domain's F2 value is kept low, as only one or two nodes around the world return any result. If such MAL domains are removed at an earlier stage in detection, examination of F2 in the A rec could potentially prove useful for distinguishing CDN domains from the other domain types. Unfortunately, in the NA rec, no such distinction between domain types exists, and F2 appears to contain no value for detection when applied to the NA rec.

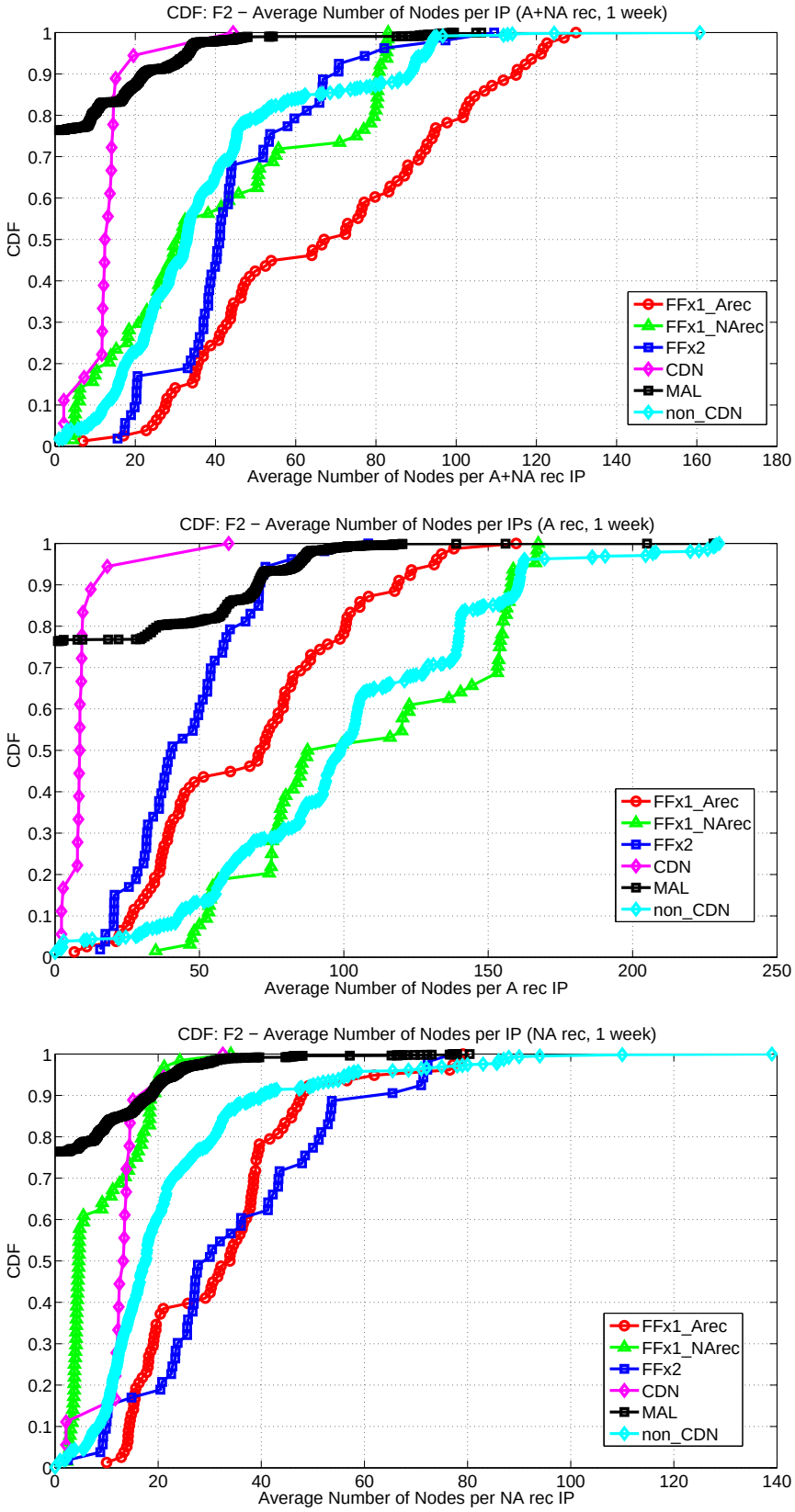


Figure 3.25: CDF of F2 - Average number of nodes per IP (1 week)

3.4.2.4 F3 - IP Overlap Between A-rec and NA-rec IPs

In Fig. 3.26, we have plotted the CDF for feature F3 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. Obviously, all the MAL domains in the figure experience IP overlap while none of the non-CDN domains do; this is because F3 is the only metric differentiating a MAL from a non-CDN domain, as discussed in Section 3.4.1.2. Amazingly, almost all of the MAL domains ($\approx$ 99%) experience 100% IP overlap, indicating that most MAL domains go for an all-or-nothing approach when it comes to advertising IPs in both record types. From the figure, we find that FFx2 domains adopt a rich and varied strategy in terms of IP overlap. While some FFx2 domains make excessive use of IP overlap (e.g., $\approx$20% have more than 70% IP overlap) and others express little to none (e.g., $\approx$15% have less than 10% IP overlap), it seems that the majority (i.e., 55%) take a moderate approach of between 10–20% IP overlap. On the other hand, 90% of FFx1_Arec domains never utilize IP overlap, and of those that do, none demonstrate more than $\approx$12% overlap. Similarly, 90% of FFx1_NArec domains never make use of IP overlap. Unlike FFx1_Arec domain, however, the FFx1_NArec domains using IP overlap do so excessively, advertising all of their limited A-rec IPs in their NA-rec as well. Lastly, we notice that there is some unexpected IP overlap observed for the CDN domains. Since our classifier only identified 18 CDN domains, the 10% of them with IP overlap corresponds to 2 *myspace* CDNs. When monitored from S. America, where they have no content or name servers, these *myspace* domains make heavy use of load-balancing over location-aware DNS strategies. For some reason, possibly due to misconfiguration, it is from this vantage point that the few overlap IPs are observed. Clearly, these anomalous instances, with less than 5% IP overlap, are insignificant when compared to the overlap observed for FFx2 and MAL domains, meaning that F3 could prove a useful feature in the detection of FFx2 domains.

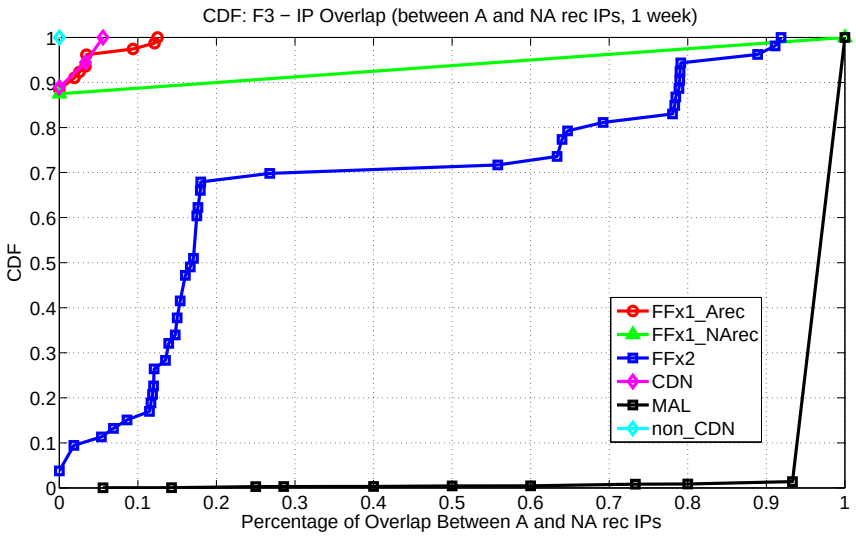


Figure 3.26: CDF of F3 - IP Overlap between A-rec and NA-rec IPs (1 week)

3.4.2.5 F4 - Percentage of IPs from Wrong Continent

In Fig. 3.27, we have plotted the CDF for feature F4 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. For comparison, we have also plotted the CDF for the percentage of IPs from the wrong *country* in Fig. 3.28. From the figures, we can see that both variations of F4 (i.e., country vs. continent) work well for differentiating CDN domains from the other domain types. However, we find that the finer-grained *country* resolution produce less impressive results, especially in the NA rec. While there appears to be some overlap with CDN and non-CDN/MAL domains in Fig. 3.27, further analysis has revealed this is the result of those offline MAL domains previously discussed in Section. 3.4.2.3, with the non-CDN domains in the plot actually being MAL domains that demonstrated no IP overlap. In this case, the 1 or 2 DNS servers that are misconfigured or returning warning web pages are either returning IPs from the same continent or bogon IPs, which are ignored in our calculations of F4 and F5 since they don't belong to a country or continent. In any case, by first identifying non-CDN and MAL domains and removing them from the set of unclassified domains, feature F4 can then prove

a useful feature for separating CDNs from FF domains. When looking at F4 in the A+NA rec, we discover that all the CDN domains have less than ≈41% of their IPs from the wrong continent, and over 95% have less than ≈17% from the wrong continent. In contrast, all the FF domains have at least ≈46% of their IPs from the wrong continent, and over 90% have more than 60% from the wrong continent. Clearly, feature F4 can prove a powerful distinguishing feature between CDN and FF domains.

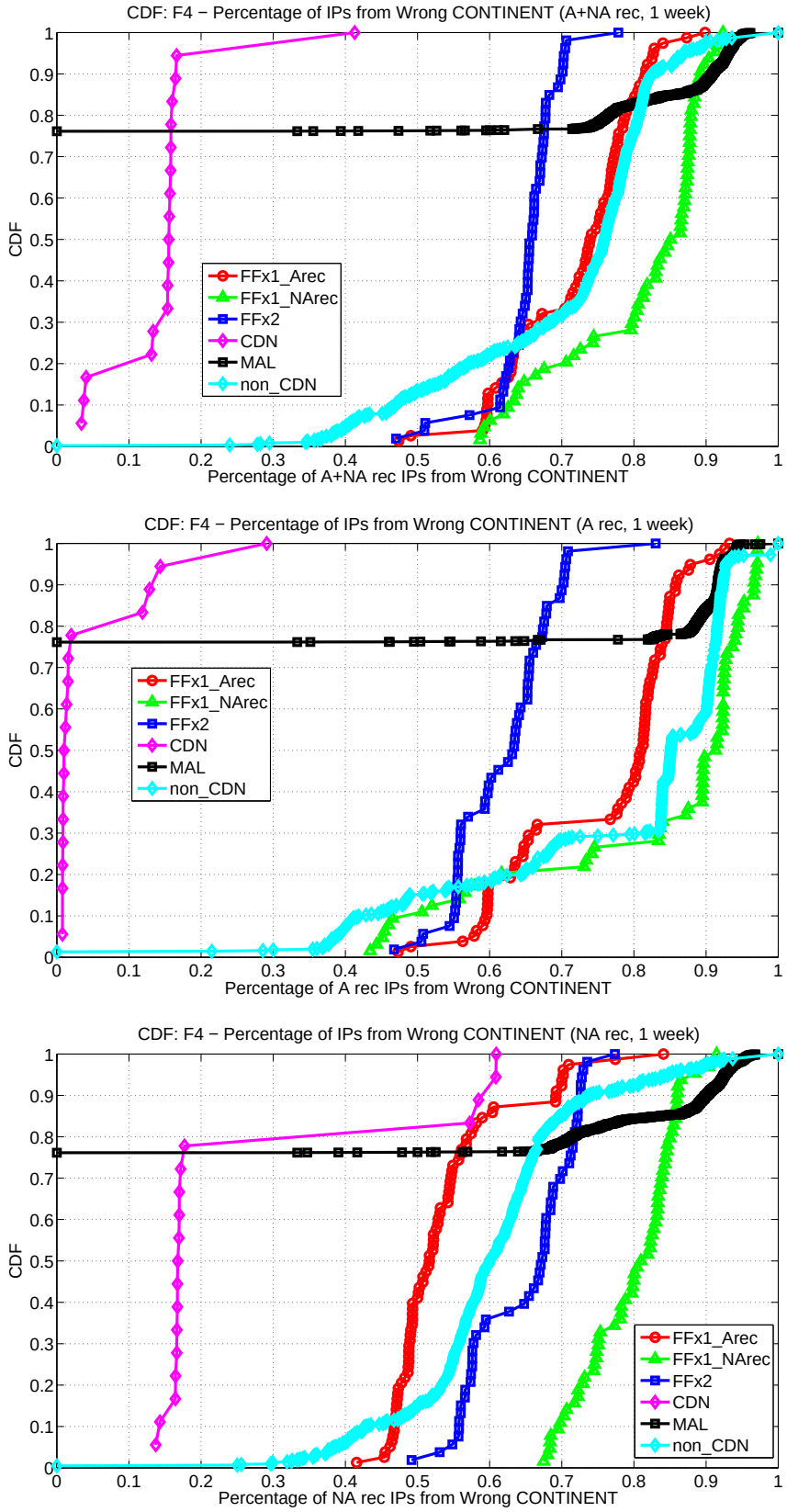


Figure 3.27: CDF of F4 - Percentage of IPs from the wrong CONTINENT (1 week)

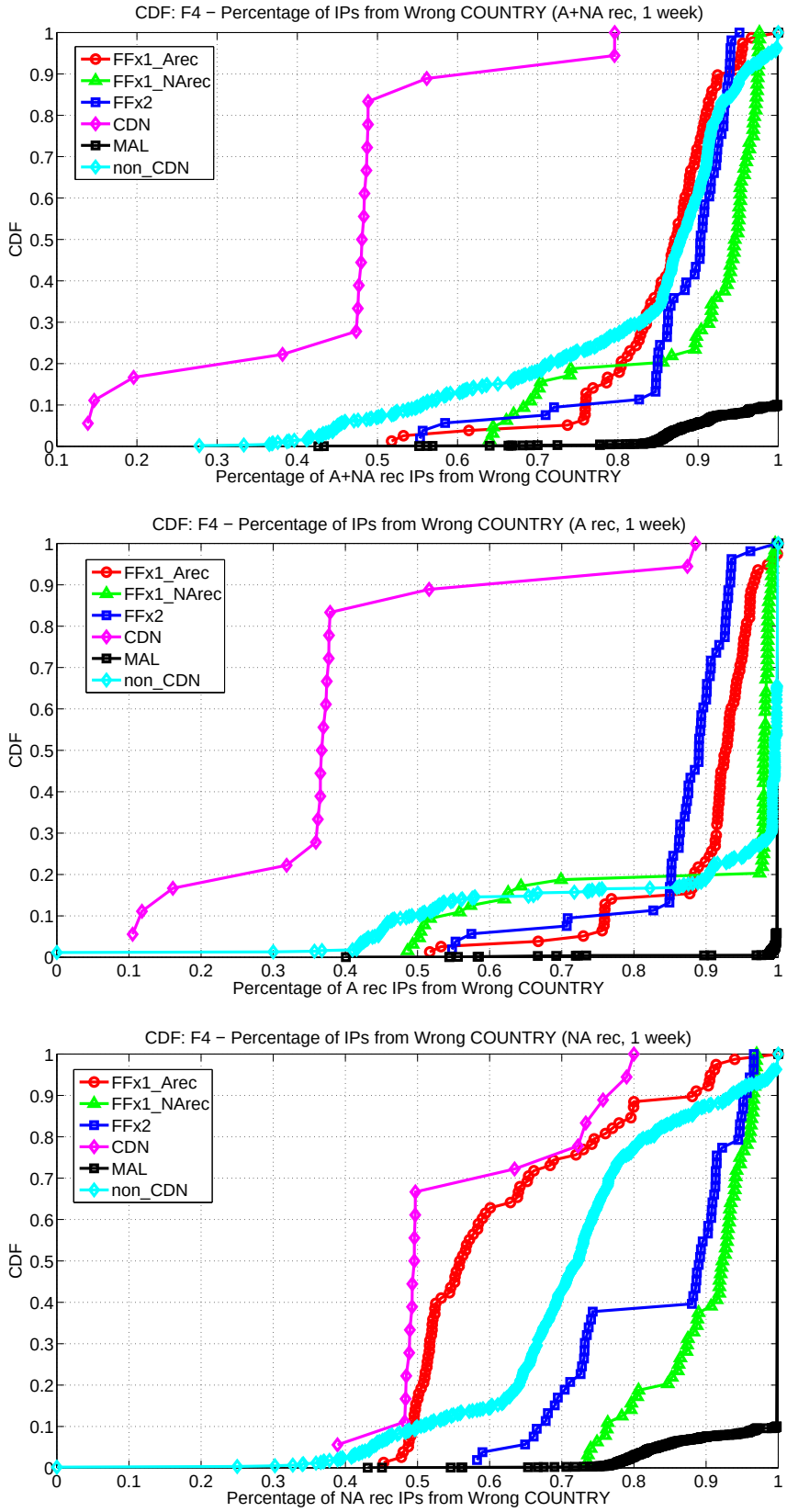


Figure 3.28: CDF of F4 - Percentage of IPs from the wrong COUNTRY (1 week)

3.4.2.6 F5 - Continental IP Distribution Average Cosine Similarity

In Fig. 3.29, we have plotted the CDF for feature F5 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. In the figure, the long-tail distribution of the non-CDN and MAL domains is the result of them actually being MAL domains which have gone offline. As previously discussed in Sections 3.4.2.3 and 3.4.2.5, these domains either return no results, bogon IPs, or IPs of warning web pages. As with feature F4, this can result in a misrepresentation of the domain. By first identifying and removing non-CDN and MAL domains from the pool of unknown domains, feature F5 can clearly be used to differentiate between CDN and FF domains. From the figure, we find a strong separation between CDN and FF domains in the A rec, while some CDNs can demonstrate values for F5 similar to FF domains in their NA rec. Deeper examination into these domains has revealed that this is an artifact of favoring a load-balancing over a location-aware advertisement strategy. This discrepancy is reduced when examining the A+NA-rec IPs, and the combination of features F5 and F4 should result in a highly accurate method of identifying CDNs from FF domains.

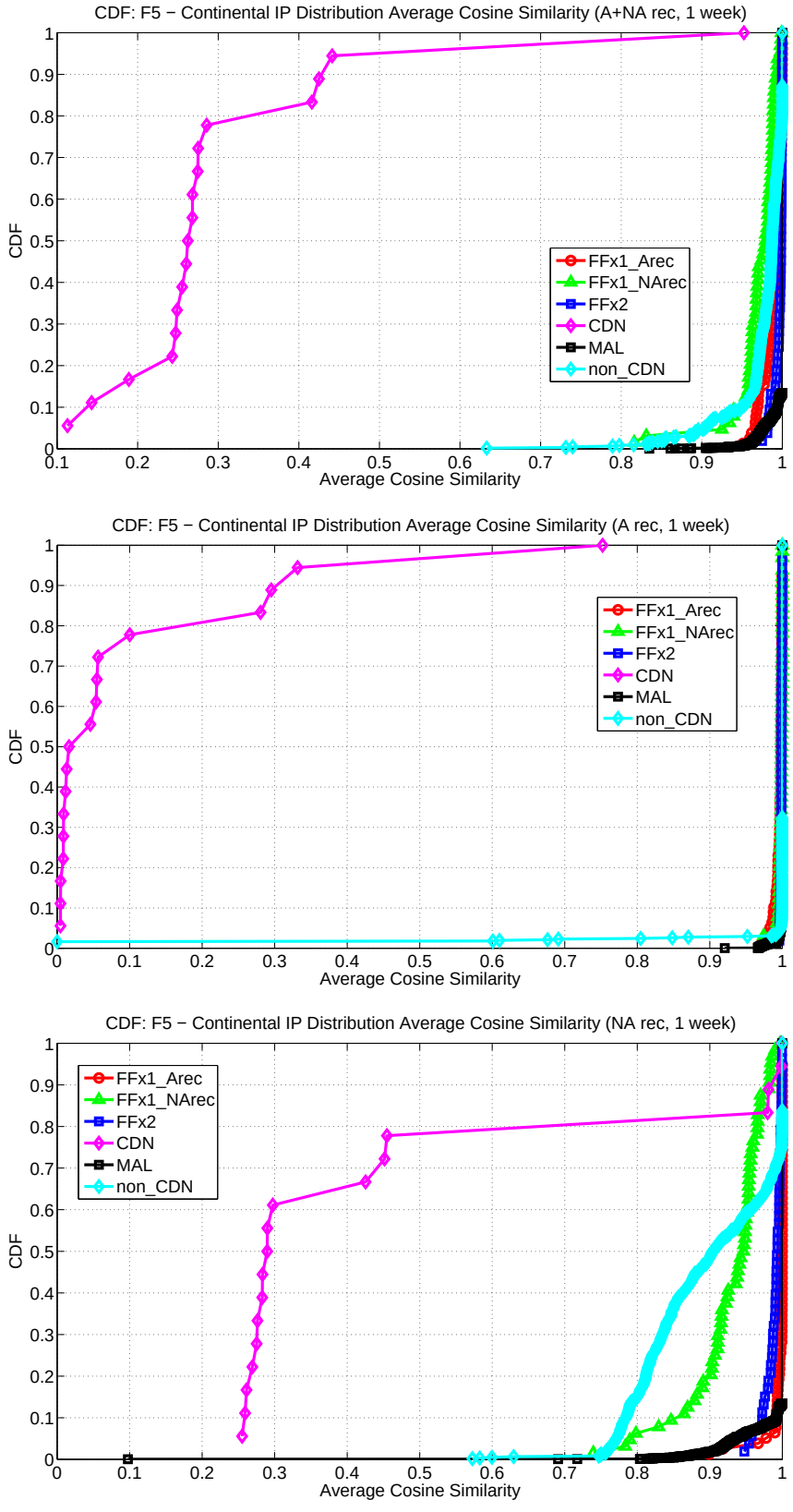


Figure 3.29: CDF of F5 - Continental IP Distribution Average Cosine Similarity (1 week)

3.4.2.7 F6 - IP Recruiting Speed

In Fig. 3.30, we have plotted the CDF for feature F6 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. There is little to say about this figure, other than that it strongly reaffirms our previous understanding that F6 does not sufficiently capture the the various IP-recruitment strategies of the different domain types. The different domain types exhibit too similar behavior, meaning that we must look elsewhere to accurately represent their various IP-recruitment behaviors.

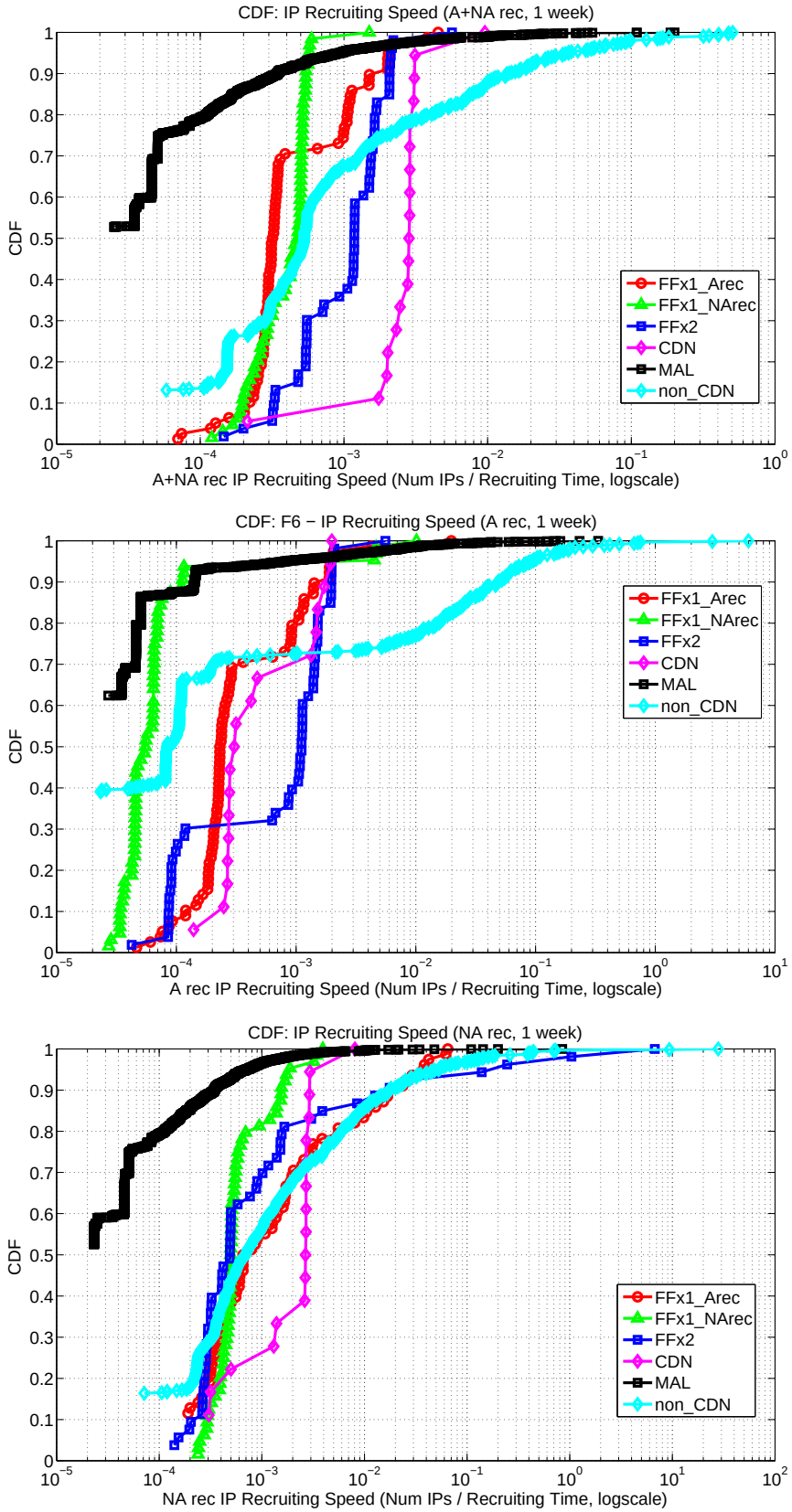


Figure 3.30: CDF of F6 - IP Recruiting Speed (1 week, log scale)

3.4.2.8 F7 - IP-Recruitment Period

In Fig. 3.31, we have plotted the CDF for feature F7 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. Like feature F6, feature F7 attempts to capture the different IP-recruitment strategies of the various domain types. Unlike feature F6, feature F7 appears to have some success in this endeavor. From the figure, we find a clear distinction between non-CDN/MAL domains and the other domain types. For example, in the A rec, all the CDN domains recruit IPs for at least 30% of their online time and all the FFx2 domains for at least 37% of their online time. All the FFx1_Arec domains recruit A-rec IPs for some percentage of their online time, with 90% of them doing so for over 50% of their online time. In contrast, less than 1% of non-CDN and MAL domains recruit A-rec IPs for more than 30% of their online time, while ≈68% don't recruit A-rec IPs at all. The FFx1_NArec domains are primarily traditional, non-CDN content servers in the A rec, possibly complimented with some bot IPs. As such, they demonstrate a slightly greater recruiting period in their A rec than the non-CDN and MAL domains, but far less than the CDN and other FF domains. For instance, more than 75% of them recruit IPs for less than 30% of their online time. However, in the NA rec, the FFx1_NArec domains have similar recruiting periods as the CDN and FFx2 domains, while the FFx1_Arec behaves like the non-CDN and MAL domains. Thus, when examining the A+NA rec, the fluxy recruiting behavior of the FFx1_Arec and the FFx1_NArec domains is sufficiently represented, aligning them with the FFx2 and CDN domains. This makes it a useful feature for identifying non-CDN and MAL domains from the other domain types. Once non-CDN and MAL domains can be identified, the process of distinguishing between CDN and FF domains using features F4 and F5 (and possibly F2 in the A rec) becomes more accurate.

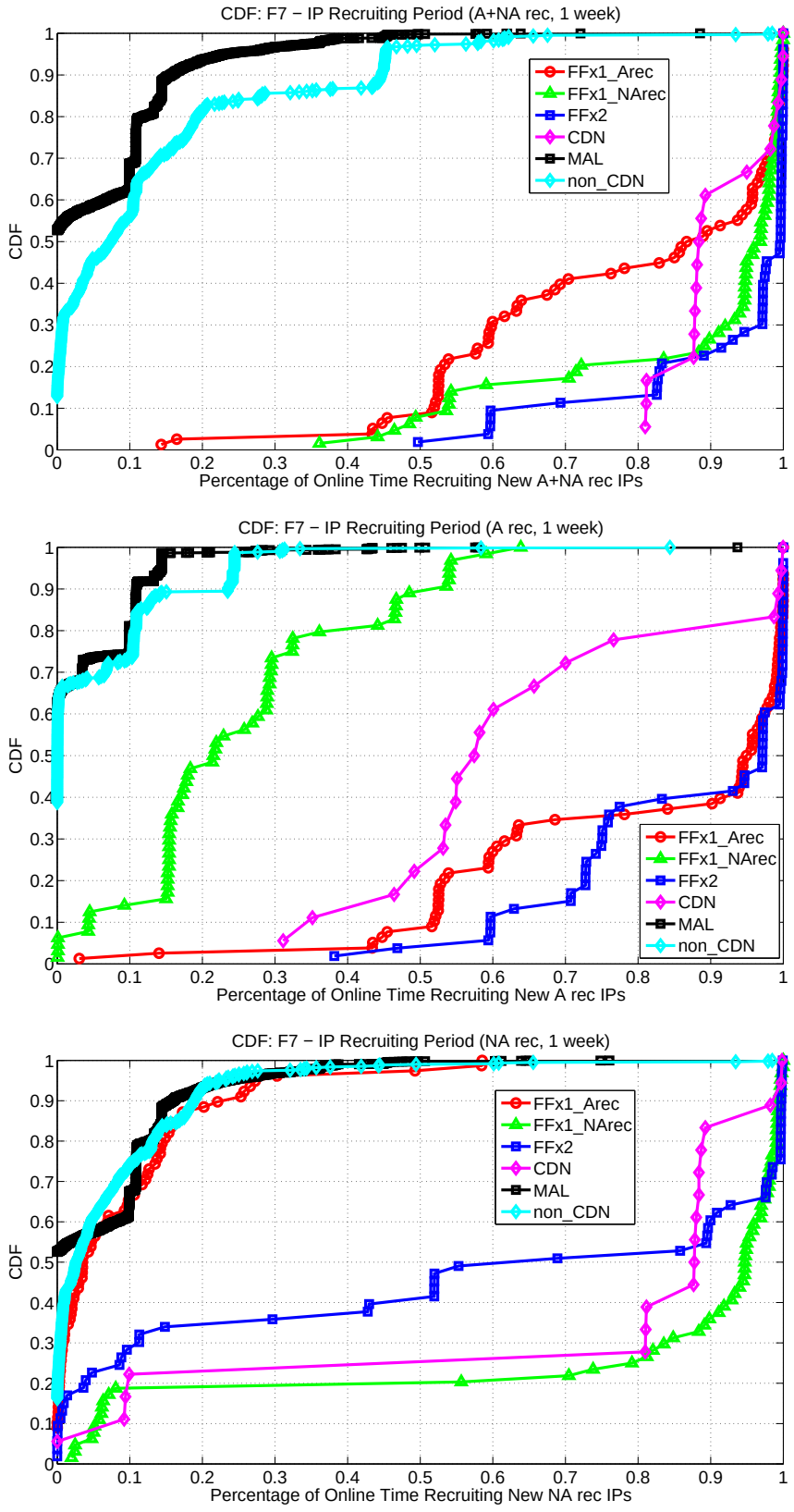


Figure 3.31: CDF of F7 - IP-Recruitment Period (1 week)

3.4.2.9 F8 - Total Unique IPs Globally

Lastly, in Fig. 3.32, we have plotted the CDF for feature F8 during a representative week for the 5,171 domains monitored by DIGGER and classified by our classifier. From the figure, it is apparent why the total number of IPs globally could aid in discriminating CDN and FF domains from non-CDN and MAL domains. In the A rec, there is a sharp separation at about 20 unique IPs globally. More than 95% of FFx2 domains and all FFx1_Arec and CDN domains have *more* than 20 unique IPs globally. Meanwhile, more than 99% of FFx1_NArec, non-CDN and MAL domains have *fewer* than 20 unique IPs globally. In the NA record, more than 75% of CDN domains and all FFx1_NArec and FFx2 domains have *more* than 30 unique IPs globally, while ≈90% of FFx1_Arec and ≈95% of non-CDN and MAL domains have *fewer* than 30 unique IPs globally. Naturally, combining them into the A+NA-rec results in an excellent separation between the non-CDN/MAL domains and the other domain types. Between ≈90–98% of CDN and FF domains have *more* than 60 unique IPs globally, while ≈99% of non-CDN/MAL domains have *fewer* than 60 unique IPs globally. It seems that this feature can prove useful in separating non-CDN/MAL domains from CDN and FF domains, provided one has a global vantage point.

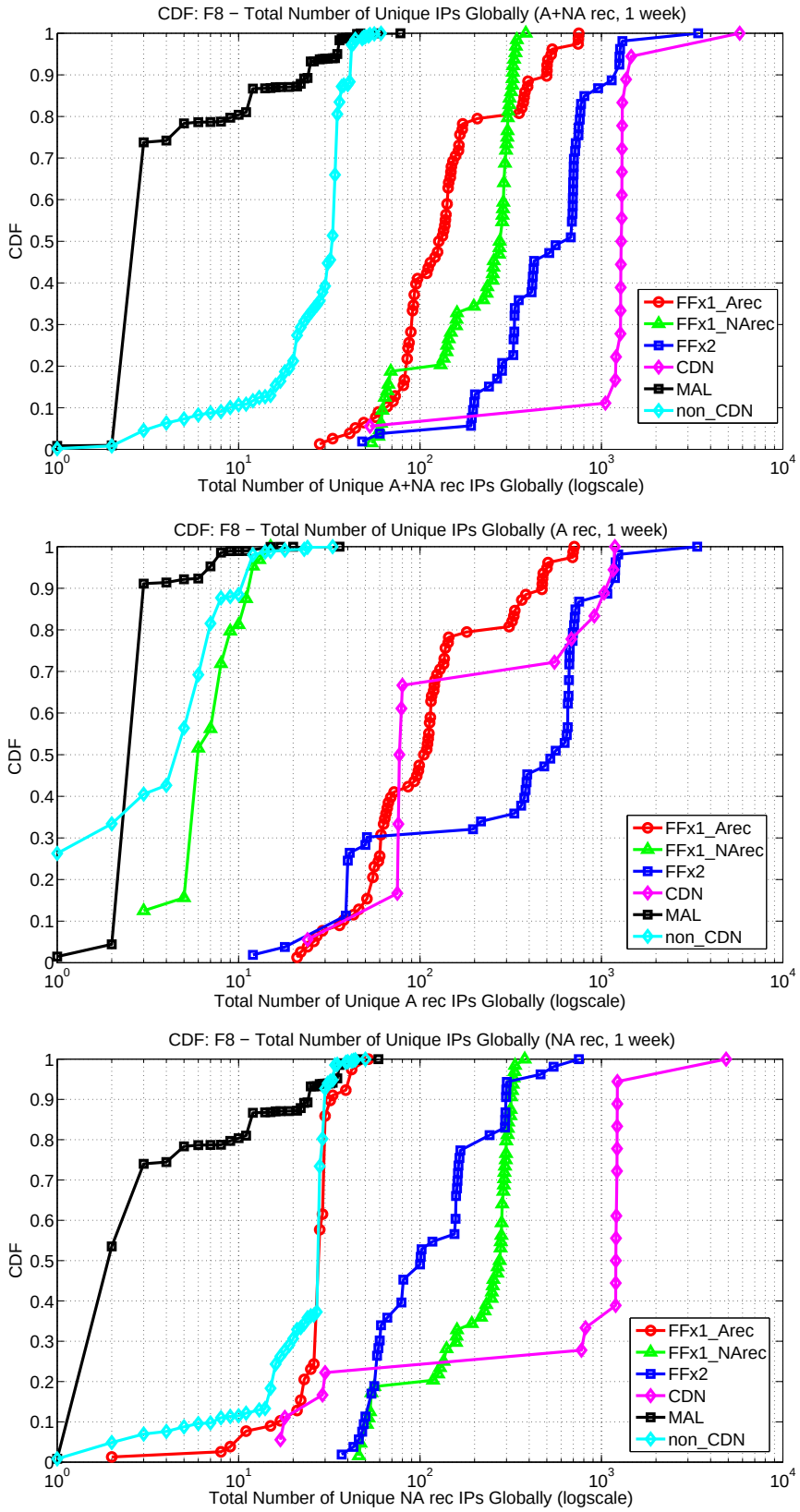


Figure 3.32: CDF of F8 - Total unique IPs globally (1 week, log scale)

3.5 Conclusion

In this chapter, we explored and analyzed the global IP-usage patterns exhibited by various types of malicious and benign domains, including FFx1 and FFx2 domains. Deploying a lightweight DNS-probing engine (DIGGER) on 240 PlanetLab nodes spanning 4 continents, we collected DNS data on a plethora of domains for over 3.5 months. Our unique global vantage point permitted the identification of several IP-usage patterns integral to the operations of the different domain types. Through detailed analysis of their publicly available DNS-query results, we were able to determine powerful distinguishing behavioral features between the domain types. Quantifying these features, we demonstrated their effectiveness for differentiation by building a multi-level, multi-week SVM classifier. The classifier proved capable of accurately discriminating between five domain types: CDN, non-CDN/MAL, FFx2, FFx1_Arec and FFx1_NArec. When applied to a set of 5,171 unknown domains, it correctly categorized them all with only a single false positive: a CDN strictly using a load-balancing DNS-advertisement strategy. Analysis of the classification results revealed the relative distribution of the various domain types in our testing data. It also granted insight into the current state of FF domains, uncovering the increased presence and versatile implementation range of FFx2 domains and evincing that fluxiness is typically more pronounced in the A recs. Lastly, we discovered an apparent trend towards using FFx1_NArec domains, which were previously unseen in the wild.

CHAPTER IV

Good Guys vs. Bot Guise: Mimicry Attacks Against Fast-Flux Detection Systems

4.1 Introduction

While their tremendous success as service networks has spurred the development of novel detection strategies, fast-flux (FF) botnets remain a persistent threat. The advent of fast, reliable FF detection systems has not yet eradicated FF botnets; rather, it has coaxed them to evolve, developing more robust, efficient, and stealthy mechanisms for subverting detection. This cycle continues, with defenders and botmasters caught in an ever-escalating "arms race". Unfortunately for the good guys, bots are free, easy to come by, capable of granting significant amounts of coordinated processing power, and incredibly effective sources of revenue.

During our global monitoring of FF botnet domains, we have observed that despite the detection mechanism or mitigation strategy imposed, botnets constantly evolve methods for subverting them, growing into ever more formidable systems—in many ways resembling enterprise-level Content Delivery Networks (CDNs). While efficient when first introduced, many FF detection systems quickly become outdated; they are designed to detect the current advertising strategies of FF botnets, which are all too easily and quickly adapted to avoid detection. Thus, it is not sufficient to

base FF detection on the current class of differentiating features. Instead, improvements could be made if botnets' limitations were also taken into consideration. While previous research has focused on identifying behavioral features uniquely intrinsic to FF botnets for detection, we have decided to take an alternate approach; assuming the role of the enemy, we explore botnets' mimicry capabilities and limitations in evading detection. Analyzing the resources currently available to FF botnets, we develop models for their bot decay, online availability, DNS-management strategies, and performance. Using these models, we examine the potential success of novel mimicry attacks against state-of-art FF detection systems, demonstrating that such attacks are easily within the capacity of current botnets. We also propose a novel spatial-detection system and evaluate mimicry attacks against it using our models. Finally, based on our findings, we introduce a new detection metric, percent connectivity, that can be used to hinder mimicry attacks. While its addition to fast-detection systems is not sufficient to catch diligent botnets using short TTLs, when applied to our proposed spatial-detection system, even the largest botnets lack the current resources necessary to evade detection over a 24-hour period. By better understanding the current arsenal botnets have at their disposal and how it can be—and is being—applied to defeat current detection strategies, we hope to foster improvements to existing systems, as well as yield new insight into the mimicry limitations of FF botnets.

4.2 Related Work

Recently, a number of techniques have been proposed to effectively detect FF domains [39, 43, 44, 56, 57]. They begin by collecting DNS queries for a large number of suspicious domains through either active or passive monitoring, over time periods ranging from 1 or 2 TTLs to weeks. From these traces, they extract similar sets of DNS features that can be used to characterize FF domains, such as the number of unique

IPs, ASes, TLDs and spatial distribution [44] of IPs.[1] Classification algorithms, such as support vector machines (SVM)[39], decision trees[56] and Bayesian network[44], are then applied to the extracted features, determining if each domain is a FF domain. In this chapter, we will demonstrate that, with the abundant resources currently available to botmasters, most of these features can be effectively subverted by the proposed mimicry attacks.

The concept of a mimicry attack was first proposed for host-based intrusion detection systems (IDSes), which typically monitor application behavior in terms of system-call sequences. Mimicry attackers attempt to slip under the radar by cloaking malicious system calls with innocuous-looking system-call sequences. Wagner and Soto [69] proposed a method that embeds nullified system-call sequences (i.e., "semantic no-ops") between malicious system calls. Kruegel *et al.* [50] devised techniques that allow an attacker to regain control after a system call by corrupting the memory and manipulating code pointers. This allows attackers to extend traditional mimicry attacks to more sophisticated IDSes. More recently, Parampalli *et al.* [55] proposed the persistent, control-flow interposition techniques that make mimicry attacks simpler, more reliable and stealthy. Similar to these previous works, in this chapter, we design and evaluate several mimicry strategies that attackers may employ to circumvent FF detection systems. The goal of our work is to anticipate attackers' next moves and better understand their capability in launching potential mimicry attacks. We hope that, by demonstrating the effectiveness of spatial-connectivity detectors against mimicry attacks and of percent connectivity as a supplemental feature in any FF botnet detection system, our work will foster the development and improvement of such techniques in future DNS-based FF detection systems.

[1]Similar to the number of ASes, the spatial-distribution feature captures the dispersive nature of FF botnet IPs.

Continent	N. America	Europe	Asia	S. America	Oceania	TOTAL
DIGGER Nodes	143	94	46	19	10	312
% of TOTAL	45.83%	30.13%	14.74%	6.09%	3.21%	

Table 4.1: Global distribution of DIGGER nodes by continent

4.3 Background

In this section, we investigate the DNS IP-advertisement patterns of current FF botnets and benign domains. First, we will describe how we set up a globally distributed DNS monitoring system. Then, we will discuss different types of domains and their unique features discovered through over 4 months of monitoring. Because most existing FF detection systems rely on DNS features to detect FF botnets, our unique, global perspective of IP-advertising strategies provides insight into the current state of FF domains and their ability to successfully evade detection via mimicry attacks.

4.3.1 Global DNS-Monitoring System

We created a distributed DNS-query engine called *DIGGER*, deployed on 312 geographically disparate nodes in the PlanetLab test bed [59]. The nodes were chosen based on the location of the DNS servers they queried, such that DIGGER would issue queries to DNS servers in different geographic locations around the world. Table 4.1 shows the continental distribution of DIGGER nodes, which is reflective of the overall distribution of available PlanetLab nodes.

DIGGER was deployed for over 4 months in early 2010 and gathered global DNS-query results for domains compiled from multiple sources, including online repositories of phishing and malware websites as well as the top 1000 most popular benign domains. On each node, for malicious and benign domains, DIGGER performed DNS queries (i.e., digs) on their A (address) records, NS (authoritative name server)

records, NA records (A records of the name servers) and the *reverse DNS* (rDNS) lookup (i.e., PTR records) for the A- and NA-record IPs. DIGGER continued to dig active domains periodically based on their observed TTL. Domains determined to be offline were dug every 24 hours to discover if they came back online.[2] For each domain, DIGGER also collected connectivity information (used to derive a bot online-decay model) on both A and NA record IPs by attempting to establish TCP connections on ports 80 and 53.[3] By applying simple heuristics on the aggregate data, we manually identified and verified 45 FF domains by looking for IP addresses with rDNS names indicative of compromised computers (e.g., dynamic, dialup).

4.3.2 Domain Types

4.3.2.1 Fast-Flux Domains

FF domains are malicious domains utilizing a FF DNS-advertisement strategy, typically built atop botnets and used for scams where the potential profits depend on the availability of the hosted services/content. To counter the unreliable connectivity of the bots hosting the malicious services/content, botmasters adopt FF DNS techniques and advertise numerous IPs in their DNS-query results with frequently-changing mappings between the domain name and different bots' IP addresses.

Figure 4.1-a illustrates the global IP usage—across all 312 DIGGER nodes—for an example FF domain. In the figure, the *Time* axis represents the time (in seconds) since DIGGER started monitoring the domain; *Node Index* represents the DIGGER node that the IP was observed on, with positive values indicating an A-record IP and negative values an NA-record IP; *IP Index* is a unique index incrementally assigned to each newly observed IP. From the figure, we notice that the FF domain slowly

[2] A domain is offline if its DNS query returns no A record.

[3] Although DNS primarily uses UDP protocol to serve requests, DNS servers also accept TCP connections on port 53 in order to support response data exceeding 512 bytes or for tasks such as zone transfer [51].

and nearly continuously advertises new, unique IPs over its entire online lifetime. Over the 4-month monitoring period, we have observed that a typical FF domain usually advertises thousands of unique IP addresses, with the most aggressive botnets advertising over 35,000. As we will demonstrate later, this huge IP pool affords botmasters considerable flexibility and abundant resources to mimic a wide range of benign DNS behaviors for evading detection.

4.3.2.2 Benign Domains

CDN domains are benign domains that use a Content Delivery Network (CDN), such as Akamai, to improve the delivery of their content. CDNs—consisting of a system of computers networked together for the purpose of improving the performance and scalability of content distribution—produce DNS-query results resembling those of malicious FF domains: numerous, changing IPs per query with short TTL values. For instance, *nfl.com*, a CDN domain shown in Fig. 4.1-b, has a short TTL (20 seconds) and constantly changes its A-record IPs, resulting in the accumulation of almost 1,200 IP addresses over all DIGGER nodes during our monitoring period. This affinity between CDN and FF domains is a consequence of their similar goal to provide reliable content delivery despite node failures as well as their shared assumption that any node can temporarily or permanently fail at any time. Consequently, it is possible for botmasters to cloak their malicious DNS-advertisement strategy as normal, benign CDN behavior. However, a CDN's DNS-advertisement profile depends on whether a location-aware[4] or load-balancing[5] strategy dominates and, therefore, can be highly dependent on the DNS server monitored. For example, most of the servers for *nfl.com* reside in N. America and Europe, with a small amount in Asia. As a result, in N. America and Europe, location-aware techniques dominate, with DNS queries consistently returning a small set of IPs belonging to the nearest servers. On

[4] Advertises IPs geographically near DNS servers to reduce transmission overhead due to distance.
[5] Advertises the IPs of servers with lower load to increase performance.

the other hand, DNS queries in continents such as S. America, where *nfl.com* has no servers, are influenced far less by location, and load-balancing techniques dominate; servers from all over N. America, Europe, and Asia are advertised based on their current load, resulting in hundreds of IPs observed per S. American DIGGER node over the monitoring period.

Non-CDN domains are benign domains that do not use a CDN for delivery of their content. Typically, non-CDN domains use a few stable content servers and a modest number of name servers. Some popular non-CDN domains may advertise more than 18 IPs in a single DNS query, using the same set of IPs in each query and rotating the order across queries for load-balancing purposes. This type of DNS strategy is often referred to as *Round-Robin DNS*.

4.4 Fast-Detection Systems (2 queries)

Given the serious threats posed by FF botnets, researchers have proposed various detection systems. In this section, we first analyze existing fast-detection systems ("good guys") and then propose mimicry strategies botmasters can use for evasion ("bad guise"). Advanced detectors and mimicry attacks against them will be described in Section 4.5.

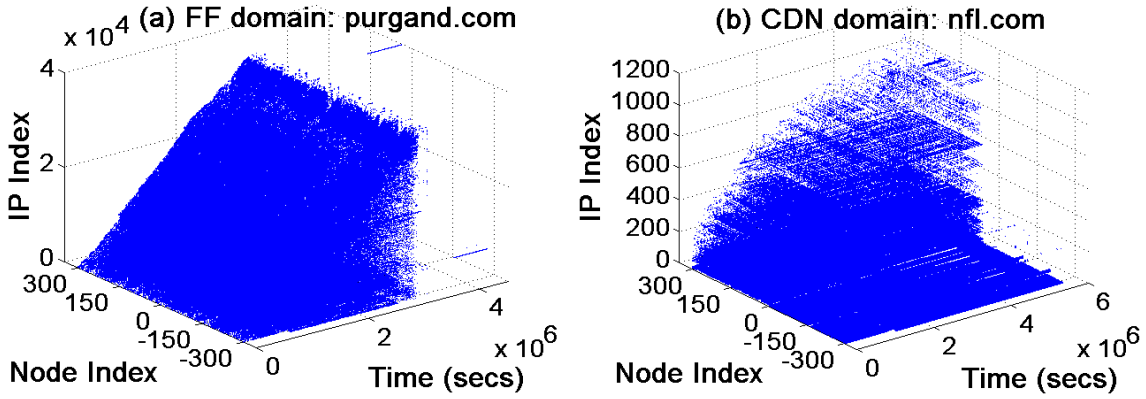


Figure 4.1: Global DNS-query results for FF and CDN domains

4.4.1 Good Guys

The original FF detection system proposed by Holz *et al.* [39] (i.e., Holz detector), shown in Eq. (4.1), and RB-Seeker's first-tier detector [43], shown in Eq. (4.2), are considered fast-detection systems, as they are capable of detecting FF domains with high accuracy from only 2 DNS queries. This is achieved through the use of a linear decision function containing weighted terms derived from the DNS queries and a bias term.

$$f(x) = 1.32 \cdot n_A + 18.54 \cdot n_{ASN} - 142.38 \tag{4.1}$$

$$\begin{aligned} f(x) = & -1.257 \cdot N_{unique\_IPs} - 26.401 \cdot N_{ASN} \\ & -13.024 \cdot N_{DNS\_bad\_words} + 162.851 \end{aligned} \tag{4.2}$$

In Eq. (4.1), the number of unique A-records IPs and Autonomous System Numbers (ASNs) are represented by n_A and n_{ASN}, respectively. In Eq. (4.2), N_{unique_IPs} represents the number of unique IPs seen in the A records, N_{ASN} the unique ASNs, and $N_{DNS\_bad\_words}$ the number of rDNS lookups containing "bad words" indicative of compromised home computers, such as comcast, dynamic, dialup, etc. In both equations, the magnitude of $f(x)$ represents the degree of confidence when classifying domain x, with positive values indicating a FF domain for Eq. (4.1) and a benign domain for Eq. (4.2).

We implemented both detectors and applied them to our manually verified set of 45 FF domains to determine how they would fare against today's FF botnet threat. They identified 6 (Holz detector) and 12 (RB-Seeker) FF domains, respectively, resulting in 86.7% and 73.3% false-negative rates and demonstrating the extent to which botnets have evolved. Both papers realize their weights and thresholds used for detection must be periodically retrained to counter future mimicry attacks. However, doing so

will increase the false-positive rate, causing benign domains—whose DNS activity is being mimicked by FF domains—to be misclassified as malicious.

4.4.2 Bot Guise

4.4.2.1 ASN-Mimicry Attack

From Eqs. (4.1) and (4.2), we find that the dominant factor in identifying FF domains is the number of unique ASNs. Clearly, an effective mimicry attack against these fast-detection systems should reduce the number of ASNs to levels seen for benign domains. Since DNS queries on benign domains often contain A-record IPs from 2 ASNs (e.g., *www.avast.com*), let us assume that a fast-detection system adopts the following overly strict policy: *over 2 DNS queries, any domain containing IPs from more than 2 ASNs will be flagged as malicious.* While this policy can result in false positives for benign domains, such as some CDNs, if it can be effectively subverted, so can more lenient constraints.

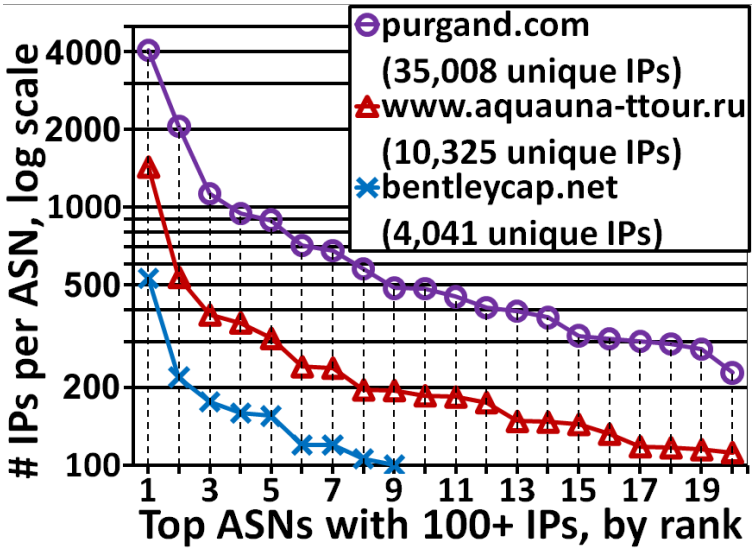


Figure 4.2: IP distribution for top 20 ASNs

To discover if this is feasible with current botnet resources, we aggregated the IPs for each FF domain globally monitored by DIGGER, determined their ASNs,

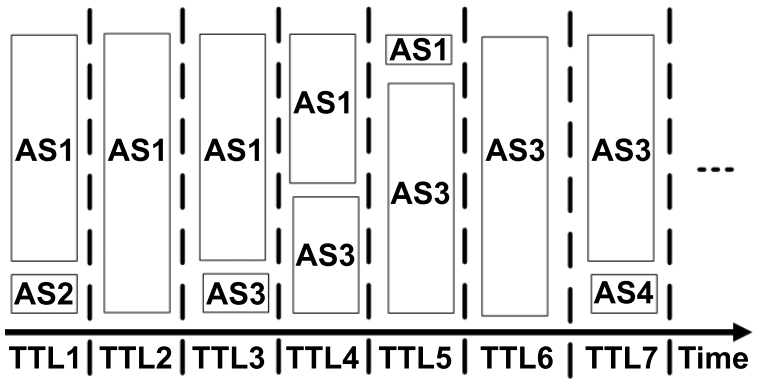


Figure 4.3: ASN-mimicry strategy (2 DNS queries)

and then analyzed their IP distribution across ASNs. We found that, despite the size of the botnet, the distribution was long-tailed, with at least one ASN containing a disproportionately large number of IPs. This is possibly due to certain ASNs (e.g., ISP networks) containing a large proportion of vulnerable computers, or from botmasters targeting certain institutions. Fig. 4.2 plots this trend for 3 representative FF domains of varying sizes; to keep the graph readable, we have only plotted the distribution for the top 20 ASNs from which the botnets have 100+ IPs.

Assuming botnets contain a suitable number of IPs from at least a single ASN (as our data indicates), there is a simple IP-advertisement strategy for mimicking the ASN behavior of benign domains. This mimicry strategy is demonstrated in Fig. 4.3, with each TTL (i.e., fresh DNS query) showing the distribution of IPs from various ASNs. For example, assume the botnet controls a large number of IPs from AS1 and a moderate to small number of IPs from AS2. At TTL1, the majority of advertised IPs are from AS1 with a smaller subset from AS2. While there exists a sufficiently large pool of online IPs from AS1, this is not the case with AS2, eventually requiring the introduction of IPs from a different ASN. However, because the detection window is 2 DNS queries, the botmaster must ensure that all the IPs seen over 2 consecutive queries belong to no more than 2 ASNs. Thus, before IPs from a new ASN can be introduced, the botmaster must first advertise only IPs belonging to one of the ASNs

present in the previous TTL, as shown in TTL2. Then, at TTL3, the botmaster is free to utilize IPs from the new ASN, AS3. If the botmaster happens to control a large number of IPs in AS3, AS1 can slowly be replaced as the dominant ASN, as shown in TTL3–TTL6. In this way, botmasters can successfully mask their use of numerous ASNs from fast-detection systems.

4.4.2.2 rDNS-Mimicry Attack

From Eq. (4.2), notice that the second most influential feature when identifying FF domains is $N_{DNS\_bad\_words}$. However, RB-Seeker asserts that a rDNS lookup on an IP will not always return a result, although when it does, it can be useful. Despite its inconsistency, the term is still an order-of-magnitude more important than the number of unique IPs. Therefore, an effective mimicry attack should include a mechanism for subverting this detection metric.

Let us assume the following aggressive detection policy: *over 2 DNS queries, any domain with more than 2 "bad words" in its rDNS results will be flagged as malicious.* Certainly, this policy is overkill, as many legitimate domains (e.g., *www.comcast.com*) will have rDNS results that contain "bad words". However, if botnets can defeat this harsh limitation, more realistic thresholds can also be subverted. If current FF botnets contain enough IPs without rDNS results (i.e., rDNS=NONE IPs), then a mimicry strategy similar to that proposed for ASNs in Section 4.4.2.1 could be applied. To determine the feasibility of this approach, we aggregated the IPs for each FF domain monitored globally by DIGGER and determined the percentage of rDNS=NONE IPs. We discovered that, for each FF domain, at least 15% of its total IPs lacked a rDNS result. Furthermore, for ≈24% of the FF domains, over 50% of their IPs lacked a rDNS result. Considering the large proportion of rDNS=NONE IPs and the fact that rDNS results for bots that aren't compromised home computers will be free of "bad words", the mimicry strategy proposed earlier for ASNs can easily be applied: IPs

without rDNS results (or without "bad words") can be used in conjunction with IPs containing "bad words", such that only 2 "bad words" are observed over 2 queries. Thus, we have confirmed the unreliable nature of rDNS, demonstrating that it can easily be subverted if used as a detection metric.

However, to be truly effective, we need to ensure that this strategy can be combined with the previous ASN-mimicry attack. Thus, for each FF domain, we analyzed the distribution of rDNS=NONE IPs across ASNs, once again observing the long-tailed distribution. This phenomenon is shown in Fig. 4.4 for 3 representative domains of varying sizes. The majority of botnets we observed still possessed enough IP-dense ASNs to sufficiently mount the dual mimicry attacks; however, to be successful, some of the smaller botnets might require more diligent maintenance.

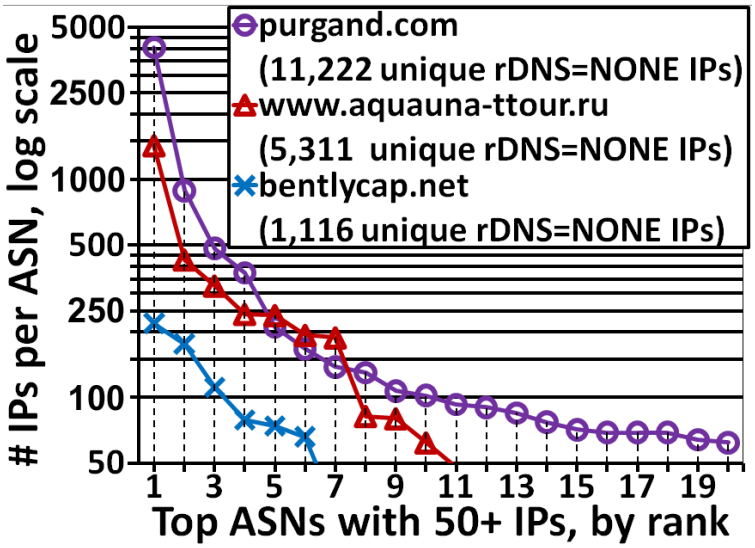


Figure 4.4: rDNS=NONE IP distribution for top 20 ASNs

4.4.2.3 IP Mimicry

Having determined that current botnet resources are capable of implementing ASN- and rDNS-mimicry attacks, we turn our attention to the final attribute utilized by the fast-detection systems in Eqs. (4.1) and (4.2), the number unique IPs. It stands

to reason that the more IPs a FF domain advertises per query, the more likely some of the bots will be online. Furthermore, because most DNS servers perform round-robin scheduling within a given TTL, advertising more IPs per query decreases the load imparted on each bot, thereby increasing the botnet's total service capacity and its revenue. Since benign non-CDN domains may also advertise a large number of stable IPs (e.g., *hostingprod.com* uses 18 IPs per DNS query), FF domains are afforded a fair amount of freedom in the number of bots they can advertise; this is supported by Eqs. (4.1) and (4.2), where the number of unique IPs is the least influential detection feature. However, non-CDN domains advertise the same set of IPs for every TTL, causing their total unique IPs to remain bounded and facilitating the use of a maximum IP threshold, N_{thresh}, for detection.

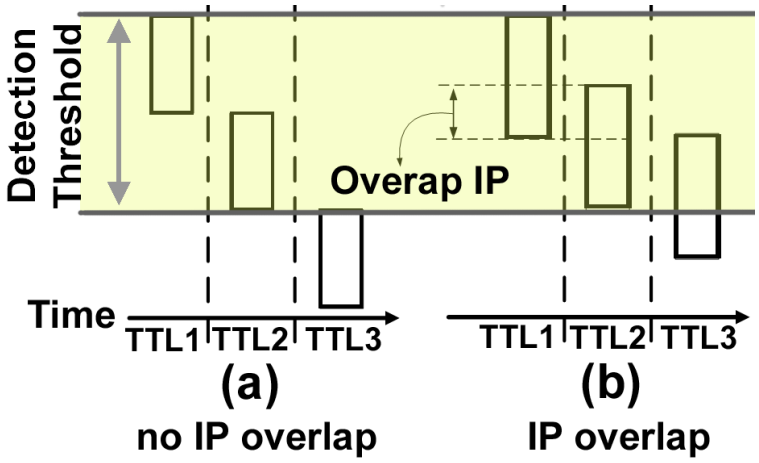


Figure 4.5: DNS IP-advertising strategies

When performing an IP-mimicry attack, there are two basic strategies for keeping the total number of IPs over two TTLs below N_{thresh}. The first, shown in Fig. 4.5-a, has no IP overlap, with the botnet advertising a completely new set of IPs every TTL. The alternate strategy, shown in Fig. 4.5-b, has IP overlap, with some of the IPs being advertised for multiple TTLs. Each strategy has certain pros and cons. Having no IP overlap allows for the rapid replacement of offline IPs; however, as can be seen from Fig. 4.5, this reduces the number of IPs that can be used for any given

TTL, which, in turn, decreases the botnet's service capacity. On the other hand, with an increase in IP overlap, more IPs can be advertised per TTL, decreasing the load per bot (from Fig. 4.5, we can see that the total number of IPs advertised per query is equal to the number of overlapped and new IPs, i.e., $N = N_{overlap} + N_{new}$); however, this reduces the rate at which offline IPs can be replaced, resulting in a greater proportion of dead bots and failed victim connections. Considering bots' unreliable connectivity, finding the optimal IP-advertisement strategy for FF domains requires a better understanding of the underlying bots' online availability (i.e., at any given time, what is the probability of bots being online).

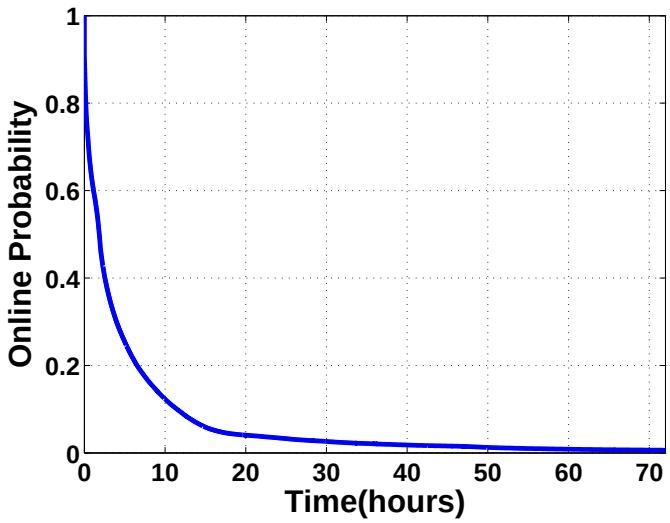


Figure 4.6: Bot online-decay model (first 72 hours)

4.4.2.4 Bot Online-Decay Model

We developed a bot online-decay model, $P_{online}(t)$, to predict the probability a bot will be online after time t. In building the model, we aggregated all the bot IPs seen for FF domains globally monitored by DIGGER, recording the time they were observed and if they were online and reachable at that time (i.e., a connection could be established). Notice that while each *individual* DIGGER node monitors the bot IPs at the granularity of the domain's TTL, they are not synchronized and

do so *independently* and at different times due to competing PlanetLab workloads. Furthermore, many botnets are used for multiple online scams; thus, many of the same bot IPs will be observed in queries on different FF domains.[6] Therefore, by combining all available data points for each bot IP—regardless of the DIGGER node's location or the FF domain it was observed for—we can build a fairly complete picture of the online times of bots currently used by FF domains. If an IP is not seen by any DIGGER node for over 12 hours, we assume that it has gone offline during that time. The resulting bot online-decay model has a long-tailed distribution, with a non-zero probability that some bots will remain online for over 2 months. In Fig. 4.6, which plots the first 72 hours of this model, the y-axis represents the probability that a bot is continuously online for more than some time t, represented by the x-axis. From the plot, it is clear that the probability of a bot being online decays exponentially with time, such that, after a day, there is less than a 10% chance it's still online. These findings reassert the notion that a bot's connectivity is highly unreliable, resulting from the varied usage patterns of the compromised computers' owners.

4.4.2.5 Performance Model

Using our online-decay model, we can determine the optimal IP-mimicry strategy in terms of performance, which we evaluate based on the number of incoming connections per unit time the botnet can handle. This metric is chosen because the revenue from FF botnets results from victims visiting the hosted content, and therefore, botmasters want to maximize the number of connections the botnet can support. If the mimicry attack drastically reduces this amount, then the bots will become overwhelmed, resulting in dropped connections and decreased revenue. We assume both the inter-arrival time of victim connections and the bots' service time are Poisson

[6]Over 11% of all FF IPs were advertised by ≈22.2% or more of the monitored FF domains; over 51% of all FF IPs were advertised by ≈6.7% or more FF domains; of the less than 34% of FF IPs advertised by only a single FF domain, over 15.7% were either dead (i.e., never reachable) or only seen once during our entire monitoring period.

processes with Markovian (i.e., exponential and memoryless) distributions, with λ and μ representing the incoming connection rate and average service times, respectively. Within a given TTL, most DNS servers perform round-robin scheduling when responding to DNS queries. As a result, incoming victim connections will be evenly dispersed among the *online* bots advertised during that TTL. Therefore at each TTL, FF botnets can be modeled as N_{online} parallel and identical M/M/1/K queues [19], where K is the online bots' queue length (i.e., the maximum connections each can queue before dropping additional connections). Applying queueing theory[19] to this model, we can calculate the *connection-loss probability* (i.e., the probability that an online bot will drop connections due to a full queue) as:

$$P_{loss} = \begin{cases} \frac{\rho^K - \rho^{K+1}}{1-\rho^{K+1}} & : \quad \rho \neq 1 \\ \frac{1}{K+1} & : \quad \rho = 1 \end{cases} \qquad \text{where } \rho = \frac{\lambda}{N_{online} \cdot \mu}$$

Because we assume that each online bot is identical, an individual bot's P_{loss} is equivalent to that of the entire botnet, allowing us to compare the various IP-mimicry attacks' performance; a higher probability of dropped connections results in fewer exploitable victims and decreased revenues.

4.4.2.6 DNS-Strategy Model

To successfully apply the performance model, we must first establish a formal relationship between an IP-mimicry attack's DNS-advertisement strategy and our online-decay model, $P_{online}(t)$, so that we can estimate the potential number of online IPs, $N_{online}(t)$, during a given TTL. This relationship is straightforward when there is no IP overlap, as in Fig. 4.5-a. Since each TTL contains a fresh set of IPs under this strategy (i.e., $N = N_{new}$), they can only decay for the time, t, that has elapsed in the current TTL; thus, for a max TTL of T_{ttl} seconds, $N_{online}(t) = N \cdot P_{online}(t)$, where $0 \leq t < T_{ttl}$.

Determining $N_{online}(t)$ becomes more complicated for a strategy utilizing IP overlap, as in Fig. 4.5-b. Because IPs are persistent for multiple TTLs, they suffer an increased probability of going offline. For modeling purposes, we must rely on the reasonable assumption that older IPs—being more likely to be offline—will always be replaced before newer IPs. Additionally, to best distribute load among their bots, we can assume that botmasters will choose to advertise as many IPs as possible without exceeding the detection threshold, N_{thresh}. These two assumptions imply an optimal replacement strategy for botmasters, from which we can deduce the following intrinsic properties: in any given TTL, (1) there exist a total of N_{new} IPs also present in the previous $0, 1, 2, \ldots, \lfloor \frac{N}{N_{new}} \rfloor - 1$ TTLs, and (2) there exist a total of (N mod N_{new}) IPs also present in the previous $\lfloor \frac{N}{N_{new}} \rfloor$ TTLs. The effect of these properties can be seen in Fig. 4.7 for two examples. Thus, for any given DNS query, we can formulate formulate $N_{online}(t)$ in terms of $P_{online}(t)$ as:

$$\begin{aligned} N_{online}(t) &= (N \bmod N_{new}) \cdot P_{online}(t + \lfloor \frac{N}{N_{new}} \rfloor \cdot T_{ttl}) \\ &\quad + \sum_{n=0}^{\lfloor \frac{N}{N_{new}} \rfloor - 1} N_{new} \cdot P_{online}(t + n \cdot T_{ttl}) \end{aligned} \tag{4.3}$$

Having defined $N_{online}(t)$ in terms of the IP-advertisement strategy, we can use it in our definition of $P_{loss}(t)$:

$$P_{loss}(t) = \frac{\rho(t)^K - \rho(t)^{K+1}}{1 - \rho(t)^{K+1}} \qquad \text{where } \rho(t) = \frac{\lambda}{N_{online}(t) \cdot \mu} \tag{4.4}$$

4.4.2.7 Evaluation of current FF botnet strategies

Before evaluating the performance of our mimicry strategies, we establish a basis for current FF botnet performance. We examine the 3 FF domains shown in Table 4.2 using our online-decay model and Eq. (4.4). We first compare these various strategies by applying $\overline{N}$ and $\overline{N}_{new}$ to Eq. (4.3), finding $\overline{N}_{online}$ when the system reaches a

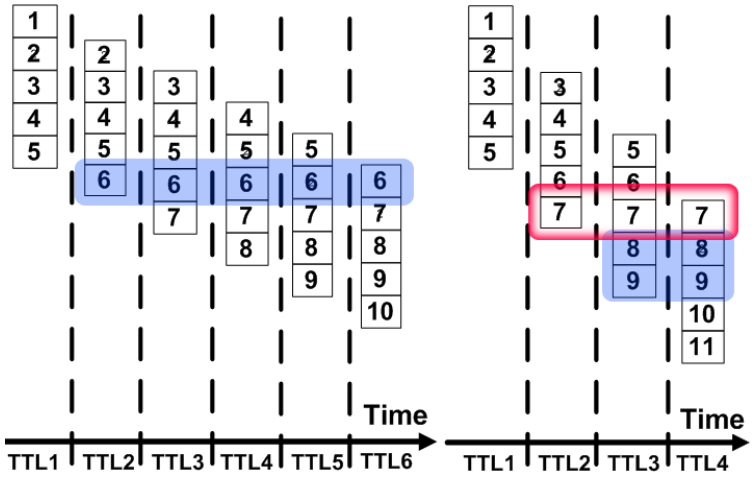


Figure 4.7: Persistence of overlapped IPs

steady state, shown in Table 4.2.[7] From the results, botmasters appear quite adept at configuring their DNS-advertisement strategies to minimize the effect of bot decay. Through the skillful manipulation of their T_{ttl}, $\overline{N}$ and $\overline{N}_{overlap}$, these remarkably different strategies were all able to achieve greater than 90% online availability (i.e., $\frac{\overline{N}_{online}}{\overline{N}}$).

Fast-Flux Domain	T_{ttl} seconds	N (# IPs per query)			$N_{overlap}$ (# of IP overlap)			$\overline{N}_{online}$ (# online IPs)	$\overline{P}_{loss}$ λ = 100 conn/s μ = 10 conn/s K = 10 (Connection Loss Prob)
		min	avg	max	min	avg	max		
old-and-girl.net	600	3	5	5	0	1	5	4.53	54.00%
bentleycap.net	300	10	10	10	0	3	10	9.41	12.10%
mountainready.com	120	4	19	20	0	14	20	17.8	0.14%

Table 4.2: Current FF DNS strategies and performance

Next, we examine the influence each type of strategy has on the botnet's overall capacity (i.e. $\hat{P}_{loss}$), which translates to the amount of potential victims and revenue. We use the values $\lambda = 100$, $\mu = 10$ and $K = 10$ for their ease of computation since, for comparison purposes, the actual choice for these values is trivial, so long as we are consistent and use the same values when evaluating each strategy. Applying Eq. (4.4) to each of the 3 botnets' DNS strategies, we determine $\overline{P}_{loss}$ once the system

[7]The values $\overline{N}$, $\overline{N}_{new}$ and $\overline{N}_{online}$ are the average values for N, N_{new} and N_{online}.

achieves a steady state, shown in Table 4.2. While the various DNS strategies offered comparable performance in terms of $\frac{\overline{N}_{online}}{\overline{N}}$, they clearly differ in the total capacity each botnet can support. This is a direct consequence of the number of bots available during a given TTL, with *mountainready.com* having approximately twice as many as *bentlycap.net* and $4x$ as many as *old-and-girl.net*. With both a large $\overline{N}$ and $\overline{N}_{overlap}$, it seems that *mountainready.com* is attempting to capitalize on the load-balancing benefits provided by a large number of advertised IPs, while using large IP overlap to keep the total number of unique IPs over 2 queries relatively low; additionally, its use of a fairly small T_{ttl} indicates a proactive approach to countering the bot-decay phenomena, which will be accentuated due to its large $\overline{N}_{overlap}$. Conversely, *old-and-girl.net* makes use of a far different strategy. With its $\overline{N}_{overlap}$ constituting only a small fraction of its $\overline{N}$, the effect of bot decay due to IP overlap is less severe, permitting less diligent IP replacement and allowing for a longer T_{ttl}. Interestingly, its decision to use a small $\overline{N}$ and $\overline{N}_{overlap}$ appears to be a double-edged sword; while keeping the total unique IPs over 2 queries low, it also results in fewer IPs per TTL for load-balancing purposes, reducing the botnet's overall capacity. Lastly, *bentlycap.net* seems to have found some middle ground between the other techniques, with a T_{ttl} and $\overline{N}$ almost exactly between the those of others. However, like *old-and-girl.net*, it has chosen a small ratio of $\frac{\overline{N}_{overlap}}{\overline{N}}$, reducing the amount of bot decay and the need for more rapid IP replacement.

4.4.2.8 IP-Mimicry Attack

Now we present how our proposed IP-mimicry attack influences the connectivity and capacity of the aforementioned FF domains. The key idea of the attack is to manipulate $\overline{N}$ and $\overline{N}_{overlap}$ such that the online time and capacity are maximized while keeping the number of IPs below the detection threshold. First, we retain the FF domains' T_{ttl} values, assuming that they were chosen by the botmasters in

response to how diligently they were willing to monitor and replace IPs. Second, in order to reduce false positives from benign, non-CDN domains advertising a large number of stable IPs, such as *hostingprod.com*, we assume—for the purposes of this mimicry attack—a detection threshold of $N_{thresh} = 20$ IPs, resulting in the following policy: *over 2 DNS queries, any domain with more than 20 unique A-record IPs will be flagged as malicious.*[8]

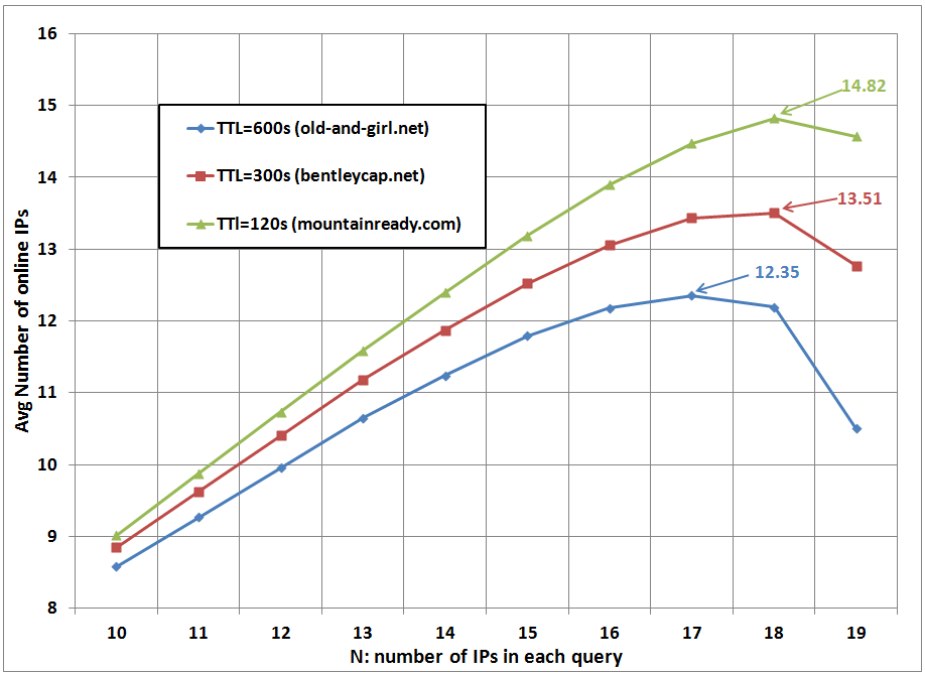


Figure 4.8: $\overline{N}_{online}$ optimization

It is clear from the results in Section 4.4.2.7, that the more online IPs available during a given TTL, the greater the botnet's overall capacity. Therefore, an optimal DNS strategy will necessarily advertise the maximum IPs allowed by the detector's threshold, N_{thresh}. This reduces the problem to an optimization problem, i.e., to determine an optimal $N_{overlap}$ that maximizes Eq. (4.5) or minimizes Eq. (4.6), subject to the constraints: $2 \cdot N - N_{overlap} = N_{thresh}$.

[8] An attack defeating the ASN and DNS-"bad word" thresholds of Sections 4.4.2.1 and 4.4.2.2 can evade detection by the Holz and RB-Seeker detectors with as many as 79 and 66 unique IPs, respectively.

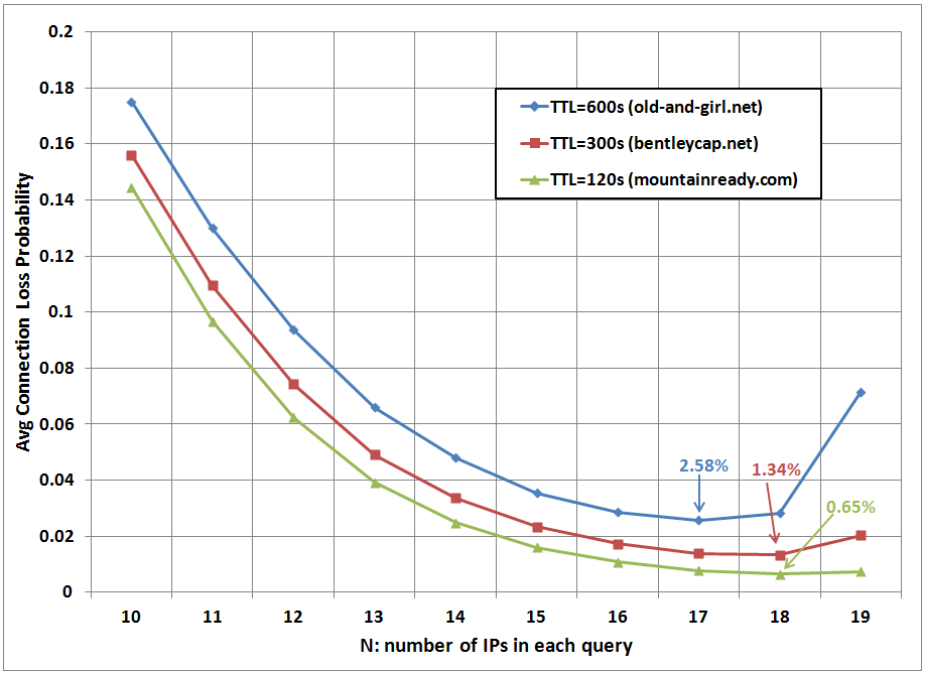


Figure 4.9: $\overline{P}_{loss}$ optimization

$$\overline{N}_{online} = \frac{\sum_{t=1}^{T_{ttl}} N_{online}(t)}{T_{ttl}} \qquad (4.5) \qquad\qquad \overline{P}_{loss} = \frac{\sum_{t=1}^{T_{ttl}} P_{loss}(t)}{T_{ttl}} \qquad (4.6)$$

For the FF domains in Table 4.2, Figs. 4.8 and 4.9 plot the results of Eqs. (4.5) and (4.6) across the search space $N \in [\lceil \frac{N_{thresh}}{2} \rceil, N_{thresh} - 1]$. From the figures, we notice while *mountainready.com* and *bentlycap.net* achieve optimal performance with $N = 18$, for *old-and-girl.net*, $N = 17$. Apparently, its longer T_{ttl} of 600 seconds results in additional bot decay, causing $N = 17$—with its 2 fewer overlapped IPs—to provide better performance. We also find that for *bentlycap.net* and *old-and-girl.net*, their $\overline{N}_{online}$ has increased to 13.52 and 12.35, while their $\overline{P}_{loss}$ has decreased to 1.34% and 2.58%, respectively. While neither of these FF domains would have been detected by the imposed N_{thresh} under their original DNS strategies, utilizing the IP-mimicry attack has kept them from being detected while also greatly increasing their performance and capacity. On the other hand, the mimicry attack caused *mountainready.com* to suffer a reduction in $\overline{N}_{online}$, dropping from 17.8 to 14.82. The attack

also caused its $\overline{P}_{loss}$ to increase from 0.14% to 0.65%. However, *mountainready.com*'s original DNS strategy advertised 24 unique IPs over 2 queries, exceeding the detection threshold. Thus, the IP-mimicry attack has allowed it to successfully evade detection with only a minor decrease in performance; its average connection-loss probability remains under 1%, and its average online IPs has been reduced by fewer than 3.

4.5 Extended-Window Detectors (more queries)

4.5.1 Good Guys

A logical extension to the fast-detection systems of the previous section is to increase the monitoring window to analyze more queries. Examining multiple TTLs when making a decision exploits a commonly known property of FF domains: they need to continuously advertise fresh IPs to account for their unstable constituent bots. While non-CDN domains may advertise a large number of IPs in their queries, they will be stable IPs that will not change over time. Thus, FF domains will quickly become exposed once additional queries are examined. Furthermore, while CDN domains can demonstrate the fluxy behavior characteristically attributed to FF botnets, for many CDNs, a longer detection window can allow their more stable nature to emerge from the chaos.

Current detectors, such as FluXOR [56] and RB-Seeker's second-tier detector, make use of longer detection windows (e.g., 1 week) to increase accuracy and support the detection of stealthy FF domains, which use slower DNS advertisement strategies to fool fast-detection systems. Like the Holz and RB-Seeker detectors, FluXOR examines the number of unique A records and ASNs. These are augmented with additional features aimed at capturing the quickly changing and dispersed nature of FF domains, such as TTL and the number of returned qualified domain names, or top-level domains (TLDs). Next, we examine how botmasters can mimic these features

to evade detection.

4.5.2 Bot Guise

4.5.2.1 ASN-Mimicry Attack

Regrettably, extending the detection window does little to weaken the ASN-mimicry attack described in Section 4.4.2.1. Because botnets seem to invariably control a sizable number of bots from within at least one ASN, the same essential attack can be performed by simply repeatedly using IPs from the same ASes to accommodate the larger detection window, as shown in Fig. 4.10.

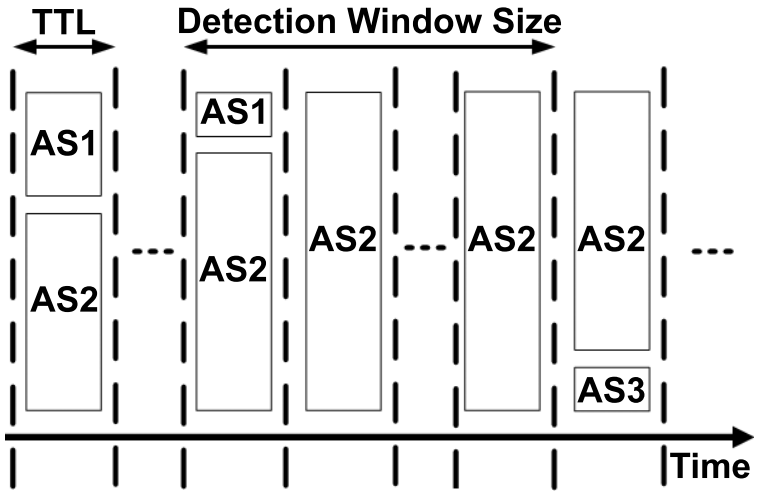


Figure 4.10: ASN-mimicry strategy (multiple queries)

4.5.2.2 rDNS-Mimicry Attack

While the specifics of FluXOR's "returned qualified domain" metric are not revealed in their paper, we can assume it operates as any TLD metric would. Essentially, for any rDNS results returned, the number of unique TLDs are calculated—the insight being that FF botnets, consisting of bots scattered across many networks, will return numerous TLDs. However, this feature also suffers from the inherent shortcoming of the rDNS lookup process, which doesn't always return a result. This allows

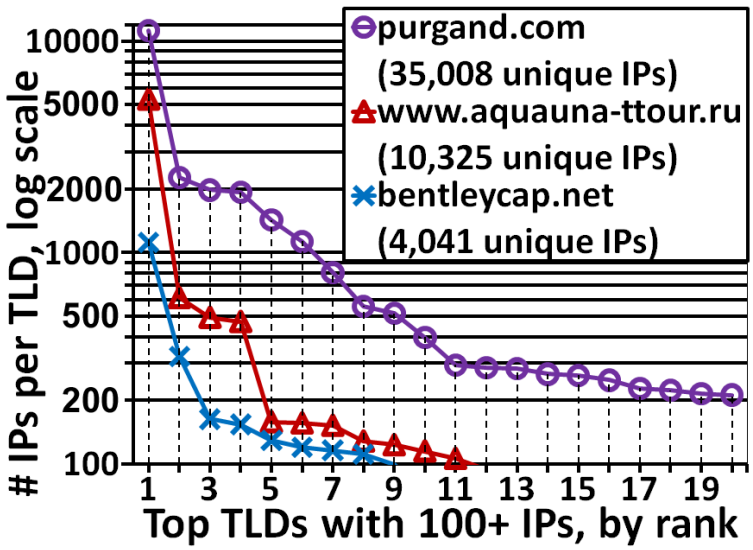


Figure 4.11: IP distribution for top 20 TLDs

for a sufficient quantity of rDNS=NONE IPs (adequately distributed across ASNs) and IPs from other TLDs to perform a similar dual-mimicry attack. Additionally, we analyzed the distribution of bot IPs across TLDs and found a similar distribution as across ASNs, in that there exist some TLDs from which a large number of bots belong. In Fig. 4.11, we have plotted this distribution for representative FF domains of varying sizes. Like the ASN distribution, it is long-tailed. While rDNS=NONE IPs dominate, there are clearly other TLDs with a sufficient number of IPs to similarly be used in the aforementioned mimicry attack, providing botmasters additional freedom in their DNS-advertisement strategies. Thus, we find that rDNS can be effectively mimicked based on FF botnets' current resources.

4.5.2.3 Improved DNS-Strategy Model

The DNS-strategy model developed in Section 4.4.2.6 can be extended to accommodate the larger detection window. First, let us assume the detector uses a detection window of D_{ttl} fresh DNS queries (where $TTL = Tttl$ seconds) and applies a threshold, N_{thresh}, on the number IPs seen during this detection window. The choice of N_{thresh} and D_{ttl} creates two scenarios botnets must overcome to avoid detection when

replacing dead IPs. In the first case, when $D_{ttl} \leq N_{thresh}$, they can simply replace at least one new IP every fresh TTL. However, if $D_{ttl} > N_{thresh}$, they can no longer introduce new IPs each TTL without exceeding N_{thresh}. To keep their total IPs below the threshold, botnets must repeat the same set of IPs over multiple TTLs before introducing any new IPs. We term this the botnet's *repetition window* and define it as R_{ttl} DNS queries (also of length T_{ttl}) for which the botnet repeats the same set of IPs, effectively extending T_{ttl}. Thus, we can determine $N_{online}(t)$ by substituting $R_{ttl} \cdot T_{ttl}$ for T_{ttl} in Eq. (4.3). The choice of R_{ttl} determines, *at most*, the amount of IP changes any detection window, D_{ttl}, will observe, represented by A_{ttl}. This relationship is shown in Eq. (4.7), and an example is given in Fig. 4.12, where we see that $A_{ttl} = 2$ when $D_{ttl} = 4$ and $R_{ttl} = 2$. Clearly, botnets can add N_{new} IPs every R_{ttl} queries, so long as Eq. (4.8) is satisfied.

$$A_{ttl} = \lfloor \frac{D_{ttl} - 2}{R_{ttl}} \rfloor + 1 \qquad (4.7) \qquad N + A_{ttl} \cdot N_{new} \leq N_{thresh} \qquad (4.8)$$

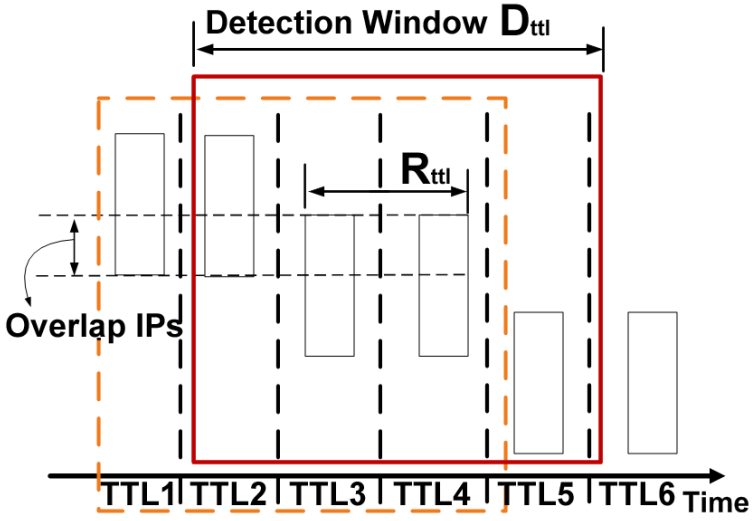


Figure 4.12: Relationship between A_{ttl}, R_{ttl} and D_{ttl}

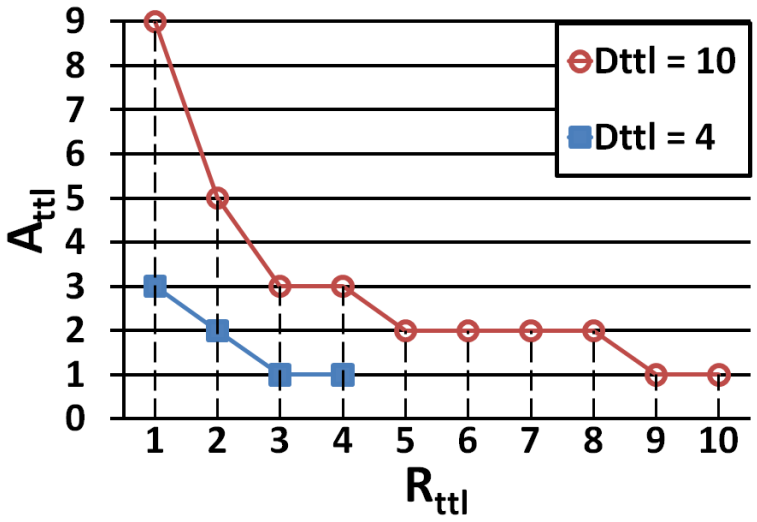


Figure 4.13: A_{ttl} when $R_{ttl} \in [1, D_{ttl}]$ and $D_{ttl} = 4, 10$

4.5.2.4 IP-Mimicry Attack (TTL-based Detection Window)

We apply the improved DNS-advertisement strategy using R_{ttl} to our previous performance model to determine how the IP-mimicry attack fares against a larger detection window. For this purpose, we examine the same real-world FF domains as in Section 4.4.2.8, once again fixing their T_{ttl} to the values originally used by each domain. Modifying Eqs. (4.5) and (4.6) to incorporate the increased detection window produces:

$$\overline{N}_{online} = \frac{\sum_{t=1}^{R_{ttl} \cdot T_{ttl}} N_{online}(t)}{R_{ttl} \cdot T_{ttl}} \quad (4.9) \qquad \overline{P}_{loss} = \frac{\sum_{t=1}^{R_{ttl} \cdot T_{ttl}} P_{loss}(t)}{R_{ttl} \cdot T_{ttl}} \quad (4.10)$$

The goal for botmasters is to determine the optimal values for R_{ttl} and N that maximize Eq. (4.9), or minimize Eq. (4.10), under the constraints that $R_{ttl} \in [1, D_{ttl}]$ and $N \in [\lceil \frac{N_{thresh}}{A_{ttl}+1} \rceil, N_{thresh} - A_{ttl}]$. The search space for N is ascertained from Eq. (4.8) and the observation that $1 \leq N_{new} \leq N$. The optimization results for $N_{thresh} = 20$ and $D_{ttl} = 4, 10$ are shown in Table 4.3. Notice that *all* the FF domains under their original DNS strategies (i.e., Table 4.2) would have been caught by the extended detection window, with the exception of *old-and-girl.net* when $D_{ttl} = 4$. However,

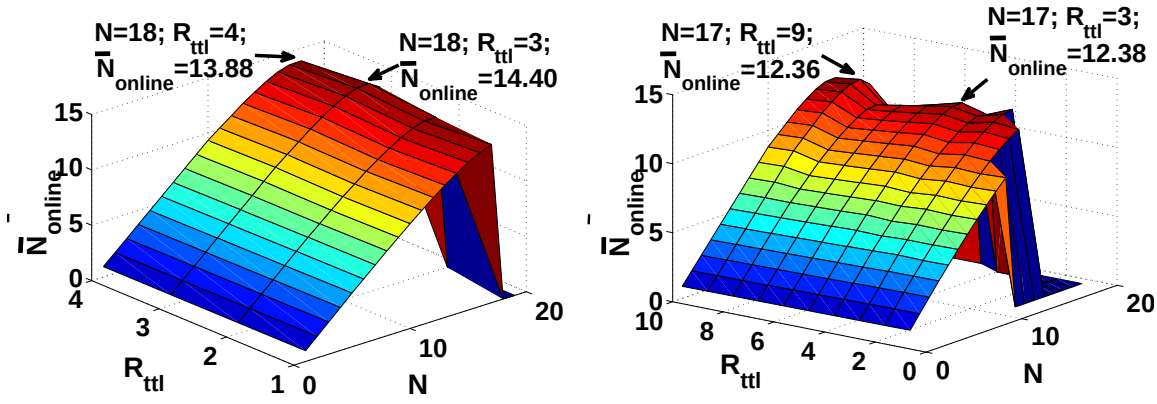
by using the proposed IP-mimicry strategy in Table 4.3, they not only successfully circumvent detection but also achieve better capacity in most cases (i.e., *bentley-cap.net* and *old-and-girl.net*). While *mountainready.com*'s performance goes down marginally, its $\overline{P}_{loss}$ is still less than 1% when $D_{ttl} = 4$ and less than 3% when $D_{ttl} = 10$. These slight decreases in performance are easily justified, considering that its original DNS strategy would have resulted in immediate detection, with 34 unique IPs seen when $D_{ttl} = 4$ and 64 when $D_{ttl} = 10$.

Parameter Setup			**Optimization Results**				
D_{ttl} # TTLs in detection win	Fast-Flux Domain	T_{ttl} seconds	N (# IPs per query)	$N_{overlap}$ (# of IP overlap)	R_{ttl} # TTLs botnet repeats IPs	$\overline{N}_{online}$ (# online IPs)	$\overline{P}_{loss}$ (λ = 100 conn/s, μ = 10 conn/s, K = 10) (Connection Loss Prob)
4	old-and-girl.net	600	16	15	1	11.20	4.92%
	bentlycap.net	300	17	16	1	12.71	3.11%
	mountainready.com	120	18	16	3	14.40	0.841%
		1	19	18	3	18.56	0.095%
10	old-and-girl.net	600	14	12	3	8.31	19.8%
	bentlycap.net	300	15	13	3	10.18	8.42%
	mountainready.com	120	17	16	3	12.38	2.58%
		1	19	18	9	18.39	0.103%

Table 4.3: Optimization Results: IP-mimicry attack against D_{ttl}

In Fig. 4.14, we show an example of the $\overline{N}_{online}$ optimization plots for *mountain-ready.com*'s $T_{ttl} = 120$, marking some local optimal points. To explain these plots, recall the relationship between D_{ttl}, R_{ttl} and A_{ttl} defined in Eq. (4.7) and shown for $D_{ttl} = 4, 10$ in Fig. 4.13. From the figures, we find that for values of R_{ttl} resulting in the same A_{ttl}, the lowest R_{ttl} is optimal. This is best exemplified when $D_{ttl} = 10$ in Figs. 4.13 and 4.14-b, with local maxima at $R_{ttl} = 1, 2, 3, 5$, and 9. To understand this behavior, recall that the amount of bot decay increases with R_{ttl} (due to repeating the same set of IPs over an extended duration). Since N and N_{new} are maximized with respect to A_{ttl} and N_{thresh} in Eq. (4.8), if A_{ttl} remains constant while R_{ttl} increases, performance will necessarily degrade, resulting in the observed trend.

As an additional experiment, we determined the optimal strategy for a FF domain using a $T_{ttl} = 1$ second, as it provides the finest granularity for adjusting the



(a) $D_{ttl} = 4$

(b) $D_{ttl} = 10$

Figure 4.14: $\overline{N}_{online}$ optimization: $T_{ttl} = 120$ seconds and $N_{thresh} = 20$

IP replacement strategy in terms of R_{ttl}. This strategy, shaded gray in in Table 4.3, achieves better results than *mountainready.com*'s original configuration. With such a short T_{ttl}, a $D_{ttl} = 10$ will only monitor the domain for 10 seconds, resulting in less bot decay and allowing for a larger N. Additionally, we find that the optimal R_{ttl} in this case is $D_{ttl} - 1$, i.e., the minimal R_{ttl} for which the detection window observes only a single IP change ($A_{ttl} = 1$). An $R_{ttl} < D_{ttl} - 1$ results in more IP changes, further limiting the maximum N achievable without exceeding the detection threshold, N_{thresh}. With an $R_{ttl} \geq D_{ttl}$, IPs are subjected to additional bot decay, while A_{ttl} remains at its minimum value of 1; regardless of R_{ttl}, eventually the detection window always observes a single IP change. Thus, at the cost of more diligent IP management, this technique maximizes the number of online IPs possible per query while minimizing the effect of bot decay.

4.5.2.5 IP-Mimicry Attack (Time-based Detection Window)

Thus far, we have defined the detection window in terms of the number of fresh DNS queries (D_{ttl}), showing that it can be subverted through the use of a repetition window, R_{ttl}. However, a detection window can also be defined in terms of absolute time (i.e., D_t seconds), requiring that the FF domain adhere to the IP threshold im-

posed over the duration D_t. Thus, the longer the duration, the more the FF domain's IPs are subjected to bot decay, decreasing performance. We model this detection technique and evaluate its susceptibility to IP-mimicry attacks under current botnet resources. For the purposes of this evaluation, we adopt a D_t equal to 1 week (as used in RB-Seeker). Certainly, requiring longer than a week to arrive at a detection decision grants botnets sufficient time to perpetrate their scams under a given domain. To find a suitable IP threshold, N_{week}, we analyzed the number of unique IPs accrued by benign CDN domains over 1 week. Not surprisingly, due to load-balancing techniques, CDN domains can advertise a large number of unique IPs depending on the DNS server monitored. For example, during a week, certain DIGGER nodes observed 171 IPs used by *nfl.com*. The amount was even greater for *www.myspace.com*, with many DIGGER nodes witnessing over 400 unique IPs weekly, and in one case, over 700. We model the IP-mimicry attacks against varying values of $N_{week} \in [100, 800]$ to evaluate how increasing the threshold—to reduce false positives—will affect a botnet's performance. To ensure that the mimicry attack would also continue to subvert fast-detection systems, we imposed the additional constraint of N_{thresh} unique IPs over *any* 2 DNS queries as before (i.e., $N + N_{new} \leq N_{thresh}$). Then, for each value of N_{week}, we calculate the maximum queries for which D_t can observe new IPs without violating N_{week} as $A'_{ttl} = \lfloor \frac{N_{week} - N_{overlap}}{N_{new}} \rfloor$. If we assume IPs are changed every TTL, then we can calculate the optimal T_{ttl} as $T_{opt} = \frac{T_{week}}{A'_{ttl}}$, where T_{week} is the number of seconds in a week. Under these constraints, the FF domain won't exceed the threshold of N_{week} unique IPs over the detection window $D_t = 1$ week. Furthermore, for any 2 queries, the number of unique IPs will satisfy the threshold N_{thresh}. Finally, notice that a repetition window, R_{ttl}, can be applied to T_{opt} to defeat a D_{ttl} detection window.

Table 4.15 shows our optimized results for N, $N_{overlap}$, and T_{opt} with $N_{thresh} = 20$ under varying thresholds of N_{week}. For all values of N_{week}, the optimal values are

$N = 19$, $N_{overlap} = 18$, and thus $N_{new} = 1$. This is because it is necessary to provide as many IPs per query as possible to counteract the enhanced botnet decay resulting from the longer T_{opt}. From the table, it is apparent that even for the strictest threshold $N_{week} = 100$,[9] the botnet will continue to have online IPs. Despite the high probability of lost connections, the botnet is still reachable and, therefore, can continue to generate revenue. For the larger thresholds of $N_{week} \geq 200$, the botnet capacity is greater than that of *old-and-girl.net* under its original configuration. These results affirm that, because benign CDN domains legitimately advertise large amounts of unique IPs over time, current botnet resources can sufficiently mount IP-mimicry attacks despite an increased detection window, D_t.

N_{week} max # unique IPs / week	T_{opt} (optimal TTL) seconds	$\bar{N}_{online}$ (# online IPs)	$\bar{P}_{loss}$ (Connection Loss Prob) λ = 100 conn/s, μ = 10 conn/s, K = 10
100	6,109	2.50	75.01%
200	3,039	4.36	56.45%
250	2,428	5.14	48.62%
400	1,515	6.88	31.71%
700	865	9.04	14.36%
800	756	9.56	11.31%

Figure 4.15: Optimization results: IP-mimicry attack against $D_t = 1$ week ($N_{thresh} = 20$: $N = 19$, $N_{overlap} = 18$)

4.5.3 Empirical Observations

We discovered several FF domains in the wild adopting some of the mimicry attacks we have presented. While the strategies employed by FF domains in the wild aren't as meticulously regular as those in our models, they are close, only deviating from their average values rarely and in small amounts. Analysis of these domains show that many of them were able to defeat the Holz and RB-Seeker detectors in

[9] A low threshold could result in many false positives since it's well below the number of IPs seen at individual DIGGER nodes for some CDN domains, such as *nfl.com* and *www.myspace.com*.

Section 4.4.1. An example FF domain is shown in Fig. 4.4, with each box in the plot representing a unique IP seen in its DNS-query results. Notice that it adds 1 or 2 IPs every ≈1,000 seconds, replacing older IPs to keep the total number equal to 5 per query; thus, it uses an $N = 5$ and an $N_{overlap} = 3, 4$. Since it has a $T_{ttl} = 10$, it's essentially using a repetition window of $R_{ttl} = 100$. Under this DNS-advertisement strategy, the FF domain can defeat a fast-detection system with an $N_{thresh} \geq 7$, as it occasionally introduces 2 new IPs per query. Furthermore, it will also defeat an extended detection window with $D_{ttl} \leq 101$ and $N_{thresh} \geq 7$. By using an average $N_{overlap} = 4$ in our model, the domain is estimated to achieve an $\overline{N}_{online} = 3.73$ (i.e., an average of 75% of its advertised IPs being online). This clearly shows that FF domains are beginning to incorporate advanced mimicry techniques to subvert detection systems, requiring novel detection methods exploiting the mimicry limitations of botnet resources. One such technique we will explore next is the use of multiple, cooperating, and geographically disparate detectors to take advantage of the spatial-mimicry limitations of botnets.

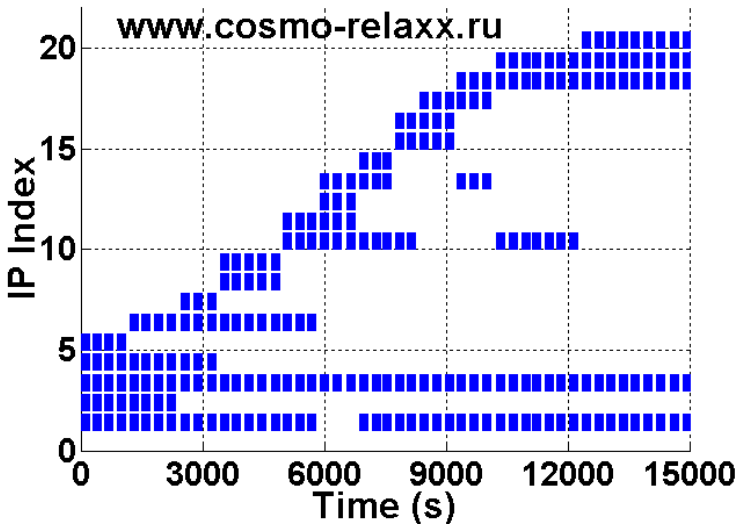


Table 4.4: Empirical observation of FF domain adopting certain evasion techniques ($T_{ttl} = 10$ seconds)

4.6 Spatial Detectors

4.6.1 Good Guys

In the previous Sections 4.4 and 4.5, we have shown that botnet resources are adequate to defeat fast-detection and extended-window detection systems. While using a week-long window and threshold can severely hurt the botnet's performance and decrease the amount of online IPs available, it is somewhat impractical, allowing FF domains to operate for a week before detection. Furthermore, because CDNs also accrue a large number of IPs during a week, the larger thresholds necessary to reduce false positives allow botnets to perform mimicry attacks with minimal reduction in online IPs and performance. It is because of these two limitations of the week-long detection window that we look to exploit the spatial properties of CDN and FF DNS-advertisement strategies. During our analysis, it became apparent that CDNs adopt a location-aware DNS-advertisement strategy, advertising IPs geographically near the queried DNS server to improve performance. CDNs use this location-aware strategy in conjunction with load-balancing techniques to further improve performance. Since botnets primarily consist of compromised home and office computers, most of them do not have 24-hour connectivity, resulting in their FF advertisement strategy. Additionally, a botnet's available bots typically follow some diurnal trend, with fewer bots available during the evening hours. This should make it more difficult for botnets to mimic the location-aware DNS-advertisement strategies of CDNs; during the evening, CDN domains will still have an ample amount of local IPs, while botnets will have to utilize online bots from other parts of the world. By monitoring DNS activity on multiple continents for 24-hours, we should be able to observe the location-aware DNS-advertisement strategy of CDNs and use it to quickly identify FF domains.

While previous sections have explored the effectiveness of mimicry attacks against real-world DNS-based FF detection systems, to the best of our knowledge, no such

system yet exists for spatial detection utilizing multiple, cooperating and geographically disparate DNS monitors. Therefore, we propose a system that makes use of the location-aware DNS-advertisement property of CDNs, evaluating if current FF botnet resources are capable of defeating it with a mimicry attack. While DIGGER was deployed on 312 nodes spanning 5 continents, this level of coverage is unrealistic for a commercial detector. Rather, we will examine a detection system utilizing 5 monitoring nodes, one on each of the continents of Asia (AS), Europe (EU), North America (NA), Oceania (OC) and South America (SA). Over every 2 queries, each node will calculate the percentage of unique IPs seen for that domain that are from a different continent, which we term the *percent wrong continent*. Since not all CDNs have complete coverage on all 5 continents, the percent wrong continent from a single vantage point will often fail as a detection metric. However, CDNs lacking coverage in certain continents will have ample, 24-hour coverage on other continents. FF botnets lack this 24-hour coverage and, therefore, wont be able to maintain, on average, the low percent-wrong-continent value in as many continents as CDN domains. Our detector makes use of this effect by calculating the *average percent wrong continent* over every 2 queries, averaging the percent wrong continent reported by the 5 monitoring nodes.

We simulate how well such a spatial detector differentiates between our manually identified FF and CDN domains using our ≈4-month DIGGER data. For the active duration of each domain, each monitoring node calculates the percent wrong continent between every pair of queries observed. Then, we pair each node's reported percent-wrong-continent results with those of 4 other nodes, one from each continent, observed in close temporal proximity, calculating the average percent wrong continent. Thus, for each FF and CDN domain, we obtain thousands of instances of the average percent wrong continent over 2 queries. We have plotted these results for our manually identified FF and CDN domains in Fig. 4.16 as a CDF. Clearly, the two different

domain types exhibit conflicting trends, with FF domains demonstrating concave growth compared to CDNs' convex growth. This bodes well for the average percent wrong continent as a detection metric against current FF and CDN domains. The plots show that for current CDN domains, the average percent wrong continent will typically be small, with over half of them having an average of 20% wrong or less. Meanwhile, no FF domains have an average less than 30% wrong; in fact, half of the FF domains have more than 80% wrong. The disparate convex and concave curves grants us an excellent 60% detection threshold, producing false-positive and false-negative rates below 5%. Next, we examine if current resources permit botnets to mimic CDN's location-aware DNS-advertisement strategy.

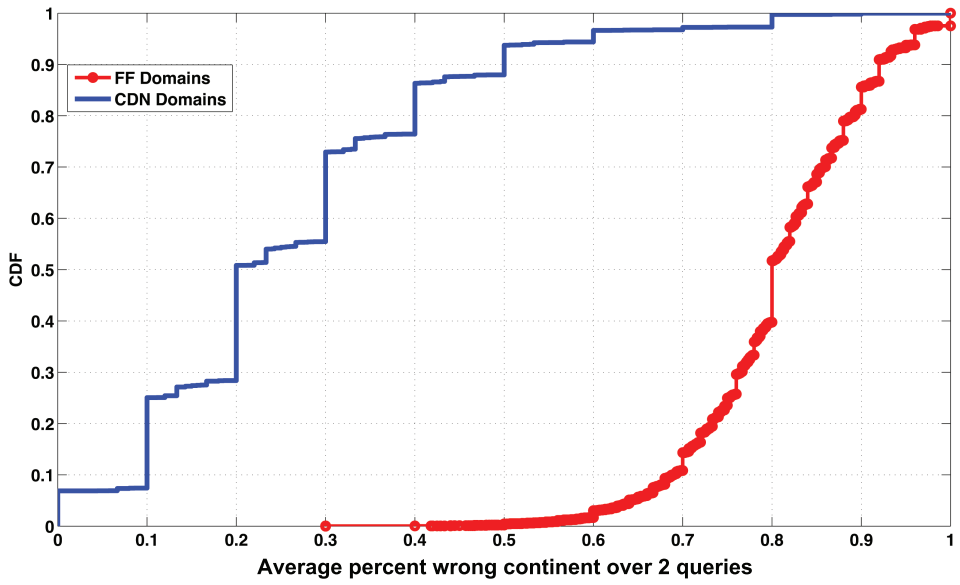


Figure 4.16: CDF of average percent wrong continent over 2 queries for FF and CDN domains

4.6.2 Bot Guise

4.6.2.1 Continental Online-Decay Models

In our spatial models, we are concerned with the available bots within a particular continent. Therefore, as we generated the global IP online-decay model in Fig. 4.6, we generate IP online-decay models specific to each continent. Additionally, we generate online-decay models for every combination of 4 continents. In this way, for each continent, we have a decay model representing IPs from that continent and a decay model for all other IPs (i.e., those not from that continent). Later, when simulating mimicry attacks against our spatial detector, we will use these continental online-decay models to reflect the botnet IP distribution specific to each continent's DNS-query results.

4.6.2.2 Diurnal Continental Online-IP Model

One of the benefits CDNs have over FF domains in providing location-aware DNS advertisement is that CDN servers are online 24-hours, while the online availability of FF domains' compromised home and office computers often follow diurnal patterns. Thus, FF domains should have difficulty mimicking location-aware advertising on each continent without hurting performance. We will test the mimicry attack capabilities of three representative FF domains of varying sizes: *purgand.com*, *www.aquauna-ttour.ru*, and *bentleycap.net*. For each domain, we find the maximum number of online IPs (i.e., bots) hourly from each continent over DIGGER's entire monitoring period.[10] Figure 4.17 demonstrates the diurnal nature of the maximum online bots per continent for the example domain *purgand.com*. We will use these maximum values later in our DNS-strategy model when modeling the percent wrong continent. These maximum values are typically higher than the average, and thus, on many days, the online bots available would be too few to support these attacks.

[10]We determine an advertised bot to be online by attempting connections on ports 80 and 53.

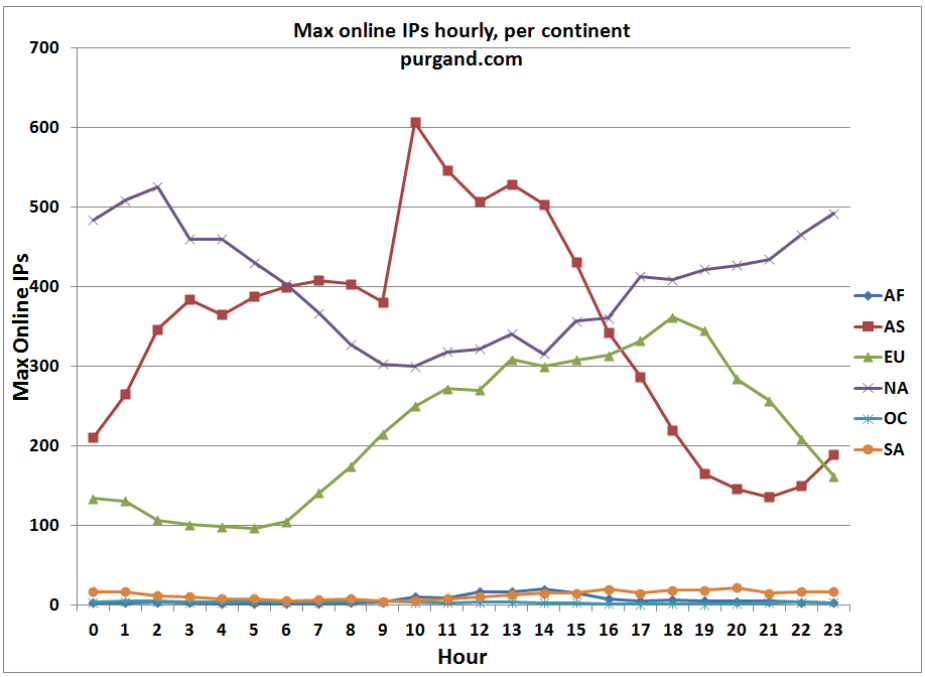


Figure 4.17: Max online IPs hourly per continent for example FF domain *purgand.com*

4.6.2.3 DNS-Strategy Model

Since our proposed spatial detector operates over 2 queries, each local monitoring node will also apply an N_{thresh} as in Section 4.4.2.3. Consequently, we will use the same DNS-strategy model as in Section 4.4.2.6 at each node, with some minor altercations. With the spatial detector, the average percent wrong continent must be below some threshold, C_{thresh}. We must assume that botmasters will attempt to advertise as many IPs as possible such that they don't exceed N_{thresh} at any given monitoring node, as this will provide the best performance for the botnet. Furthermore, during hours when there are not enough online bots for a given continent, botmasters will use online bots from other continents. Lastly, botmasters must ensure that the average percent wrong continent is below C_{thresh}. If necessary, botmasters will insert fake (and thus offline) bots into the DNS-query results to bring down the average; we assume that fake IP insertion is applied equally to all continents until the threshold has been reached.

Since we are utilizing 5 cooperating monitoring nodes, this model actually consists of 5 complementary models, one for each node (i.e., continent). For each node, we determine the amount of online bots available per hour as in Section 4.6.2.2, which we term N_{local}. When a node's continent doesn't contain enough online bots to support 2 DNS queries, we assume botmasters will advertise bots from other continents, which we term N_{other}, and they will be impartial about which continents the advertised bots reside in. Therefore, for each monitoring node, $N = N_{local} + N_{other}$. After determining N_{local} and N_{other} for each node, we can find each node's percent wrong continent as $\frac{N_{other}}{N}$. If the average percent wrong continent for an hour exceeds the threshold C_{thresh}, then fake IPs from the same continent as the monitoring node are inserted to bring down the average, which we term N_{fake}. To keep the average down, these fake IPs are always advertised, yet they are never online, essentially lowering that node's N_{thresh} by N_{fake}. Therefore, they are not included when calculating a node's N_{online}, as $N = N_{local} + N_{other}$.

Having added fake IPs to defeat C_{thresh} if necessary, we can finally model each node's N_{online}. Let the continental online-decay models, as described in Section 4.6.2.1, for N_{local} and N_{other} be represented by P_{local} and P_{other}, respectively. Then, we find N_{online} as in Eq. (4.3), both for P_{local} and P_{other} in place of P_{online}. Let L_{online} represent the number of online bots found using the P_{local} online-decay model, and let O_{online} represent the number of online bots found using P_{other}. Let the percentage of bots from the node's correct continent be represented by $p = \frac{N_{local}}{N}$. Then for a given node, $N_{online} = p * L_{online} + (1 - p) * O_{online}$. Finally, P_{loss} can be found for each node using Eq. (4.4).

4.6.2.4 Spatial-Mimicry Attack

We evaluate the hourly mimicry attack capabilities against our proposed spatial detector for 3 FF domains of representative sizes. We use the continental online-IP

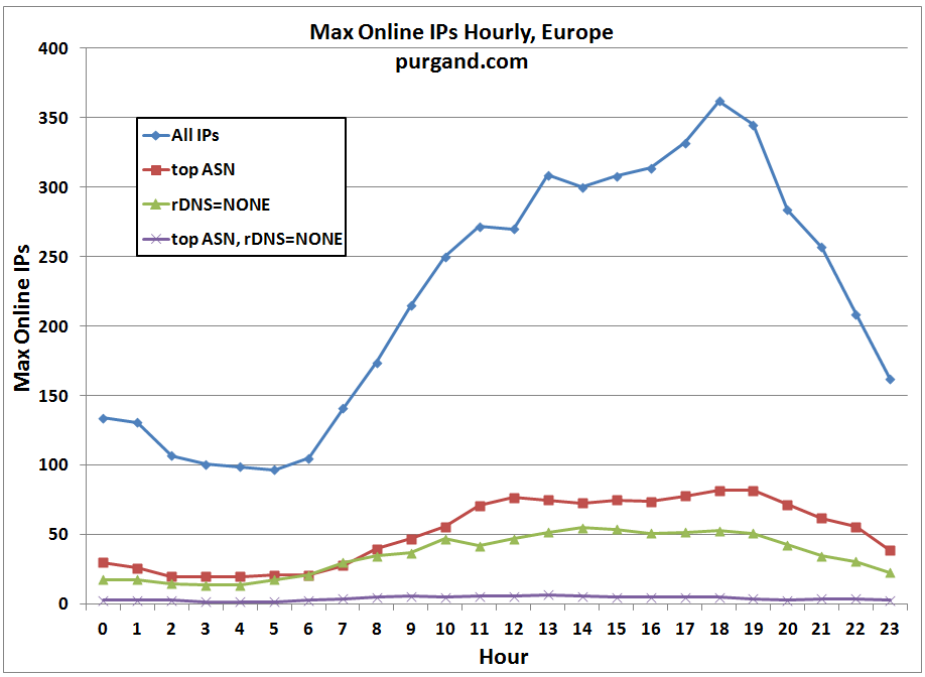


Figure 4.18: Max online IPs hourly in Europe for example FF domain *purgand.com* under various ASN/rDNS constraints

models described in Section 4.6.2.2 to determine the botnets' hourly online resources. In addition, we evaluate how further constraints on the number of unique ASNs and rDNS results influence the mimicry attack. We achieve this by filtering the maximum online IPs seen per hour as in Section 4.6.2.2, such that all IPs are from a single ASN (i.e., the top ASN), all IPs have rDNS=NONE, or all IPs are both from the top ASN and have rDNS=NONE. Notice, if each continent only uses IPs from the top ASN, then IPs used from other continents can belong to a second ASN without violating a threshold of 2 ASNs per query. Likewise, if all IPS have rDNS=NONE, then a threshold on rDNS results returned will not be violated. As can be expected, the addition of each constraint reduces the amount of online bots available to the botmaster. This can be seen in Fig. 4.18, which shows the constraints' effects on the *purgand.com*'s hourly online bots in Europe.

We model the DNS-advertisement strategy as in Section 4.6.2.3, using an $N_{thresh} = 20$ and a $C_{thresh} = 60\%$, as discovered in Section 4.6.1. This results in the following

detection policy: *over 2 DNS queries, any domain with more than 20 unique IPs observed at any individual monitoring node or with an average percent wrong continent of more than 60% across all nodes will be flagged as malicious.* From our analysis, we find that most current FF domains don't possess the hourly resources necessary to maintain the location-aware DNS-advertisement strategy needed to defeat our spatial detector. Only the larger botnets, such as *purgand.com*, have sufficient hourly online IPs in enough different continents to do so for modest T_{ttl} values. For most hours, the majority of FF domains have enough online IPs to support only a few fresh queries per hour. However, since we assume the online bots we observe in an hour are online for the entire hour, we will also assume that if a domain has sufficient resources to satisfy 2 DNS queries, as described in our DNS-strategy model, then it can maintain that DNS-advertisement strategy during the entire hour. This assumption over-represents the botnets' actual online resources according to our measurements, however, it represents what a spatial detector, taking a single, unbiased snapshot for detection at a random point of time within that hour, might observe for the botnets.

Using our hourly continental online IP-models for various constraints, we model the hourly DNS-advertisement strategy for each continent's node as outlined in Section 4.6.2.3 for the example domains. Using the three common T_{ttl} values of 120, 300 and 600 seconds, we choose values for $N_{overlap}$ such that $2*N-N_{overlap} = N_{thresh}$, optimizing Eqs. (4.5) and (4.6) as before. Thus, we obtain an optimal DNS-advertisement strategy for each domain, continent, T_{ttl}, ASN/rDNS constraint and hour combination, shown in Table. 4.19 for the optimal hours. For a visual representation, we show the optimal hours' N_{online} and P_{loss} for *bentleycap.com*'s continents using a 600-second T_{ttl} under various ASN/rDNS constraints in Figs. 4.20 and 4.21, respectively. From the table, we find that the use of a smaller T_{ttl} increases the number of online IPs and improves performance since it reduces the amount of IP decay. We also notice that the ASN/rDNS constraints reduce the online availability and hinder performance.

While most current FF botnet resources are inadequate for supporting the short T_{ttl} values we examine, we have shown previously, with our week-long detection window, that even the decay resulting from longer T_{ttl} values can still support some botnet functionality. When taken in conjunction with these optimal FF domain results, it appears that current botnet resources are capable of mimicking location-aware DNS-advertisement strategies to evade detection by our spatial detector.

Domain	Continent	TTL=600s				TTL=300s				TTL=120s			
		All IPs	top ASN	rDNS=NONE	top ASN, rDNS=NONE	All IPs	top ASN	rDNS=NONE	top ASN, rDNS=NONE	All IPs	top ASN	rDNS=NONE	top ASN, rDNS=NONE
purgand.com	AS	**12.26 , 02.72%**	12.26 , 02.72%	12.26 , 02.72%	**12.26 , 02.72%**	**13.43 , 01.40%**	13.43 , 01.40%	13.43 , 01.40%	**13.43 , 01.40%**	**14.81 , 00.65%**	14.81 , 00.65%	14.81 , 00.65%	**14.81 , 00.65%**
	EU	**12.05 , 03.06%**	12.05 , 03.06%	12.05 , 03.06%	**11.13 , 05.08%**	**13.21 , 01.59%**	13.21 , 01.59%	13.21 , 01.59%	**12.21 , 02.77%**	**14.60 , 00.73%**	14.60 , 00.73%	14.60 , 00.73%	**13.32 , 01.48%**
	NA	**12.52 , 02.34%**	12.52 , 02.34%	12.52 , 02.34%	**11.71 , 03.68%**	**13.64 , 01.24%**	13.64 , 01.24%	13.64 , 01.24%	**12.84 , 01.95%**	**14.82 , 00.65%**	14.82 , 00.65%	14.82 , 00.65%	**14.06 , 00.98%**
	OC	**12.21 , 02.79%**	12.14 , 02.91%	12.29 , 02.68%	**11.66 , 03.79%**	**13.38 , 01.44%**	13.31 , 01.50%	13.45 , 01.39%	**12.79 , 02.01%**	**14.75 , 00.67%**	14.71 , 00.69%	14.79 , 00.66%	**14.03 , 01.00%**
	SA	**14.98 , 00.59%**	13.08 , 01.71%	12.72 , 02.10%	**12.31 , 02.64%**	**15.76 , 00.39%**	14.22 , 00.90%	13.86 , 01.10%	**13.40 , 01.42%**	**16.83 , 00.22%**	15.36 , 00.48%	15.09 , 00.56%	**14.48 , 00.78%**
www.aquauna-ttour.ru	AS	**12.26 , 02.72%**	11.69 , 03.73%	12.26 , 02.72%	**09.82 , 10.00%**	**13.43 , 01.40%**	12.82 , 01.98%	13.43 , 01.40%	**10.80 , 06.02%**	**14.81 , 00.65%**	14.06 , 00.98%	14.81 , 00.65%	**11.78 , 03.52%**
	EU	**12.05 , 03.06%**	11.17 , 04.97%	11.61 , 03.90%	**09.96 , 09.30%**	**13.21 , 01.59%**	12.25 , 02.71%	12.73 , 02.07%	**10.90 , 05.70%**	**14.60 , 00.73%**	13.35 , 01.46%	13.98 , 01.03%	**11.88 , 03.33%**
	NA	**12.47 , 02.42%**	11.08 , 05.22%	11.70 , 03.72%	**09.84 , 09.89%**	**13.60 , 01.27%**	12.17 , 02.84%	12.82 , 01.97%	**10.81 , 05.99%**	**14.82 , 00.65%**	13.30 , 01.50%	14.06 , 00.98%	**11.79 , 03.50%**
	OC	**12.21 , 02.79%**	11.60 , 03.94%	11.73 , 03.65%	**09.81 , 10.02%**	**13.38 , 01.44%**	12.73 , 02.08%	12.85 , 01.94%	**10.79 , 06.07%**	**14.75 , 00.67%**	13.99 , 01.02%	14.06 , 00.98%	**11.77 , 03.55%**
	SA	**14.98 , 00.59%**	13.26 , 01.54%	11.95 , 03.23%	**09.87 , 09.72%**	**15.76 , 00.39%**	14.17 , 00.93%	13.07 , 01.72%	**10.83 , 05.94%**	**16.83 , 00.22%**	15.22 , 00.52%	14.23 , 00.89%	**11.80 , 03.48%**
bentleycap.net	AS	**12.31 , 02.64%**	11.12 , 05.10%	10.44 , 07.29%	**08.67 , 16.80%**	**13.47 , 01.37%**	12.20 , 02.79%	11.50 , 04.13%	**09.44 , 11.93%**	**14.82 , 00.65%**	13.30 , 01.50%	12.52 , 02.32%	**10.38 , 07.51%**
	EU	**12.05 , 03.06%**	10.99 , 05.46%	09.85 , 09.86%	**08.72 , 16.51%**	**13.21 , 01.59%**	12.08 , 02.99%	10.81 , 06.01%	**09.48 , 11.70%**	**14.60 , 00.73%**	13.22 , 01.57%	11.78 , 03.52%	**10.42 , 07.34%**
	NA	**12.52 , 02.34%**	11.15 , 05.01%	10.46 , 07.23%	**08.63 , 17.10%**	**13.64 , 01.24%**	12.23 , 02.75%	11.50 , 04.11%	**09.43 , 12.03%**	**14.82 , 00.65%**	13.29 , 01.51%	12.52 , 02.32%	**10.36 , 07.57%**
	OC	**11.99 , 03.17%**	11.04 , 05.32%	10.40 , 07.48%	**08.61 , 17.27%**	**13.22 , 01.58%**	12.13 , 02.90%	11.46 , 04.21%	**09.41 , 12.14%**	**14.64 , 00.71%**	13.27 , 01.53%	12.50 , 02.36%	**10.34 , 07.65%**
	SA	**13.08 , 01.71%**	11.20 , 04.88%	10.46 , 07.22%	**09.27 , 12.97%**	**14.22 , 00.90%**	12.29 , 02.66%	11.51 , 04.11%	**10.13 , 08.52%**	**15.36 , 00.48%**	13.37 , 01.44%	12.53 , 02.32%	**11.09 , 05.14%**

Figure 4.19: Optimization results: IP-mimicry attack against 2-query spatial detection ($N_{thresh} = 20$, $C_{thresh} = 60\%$)

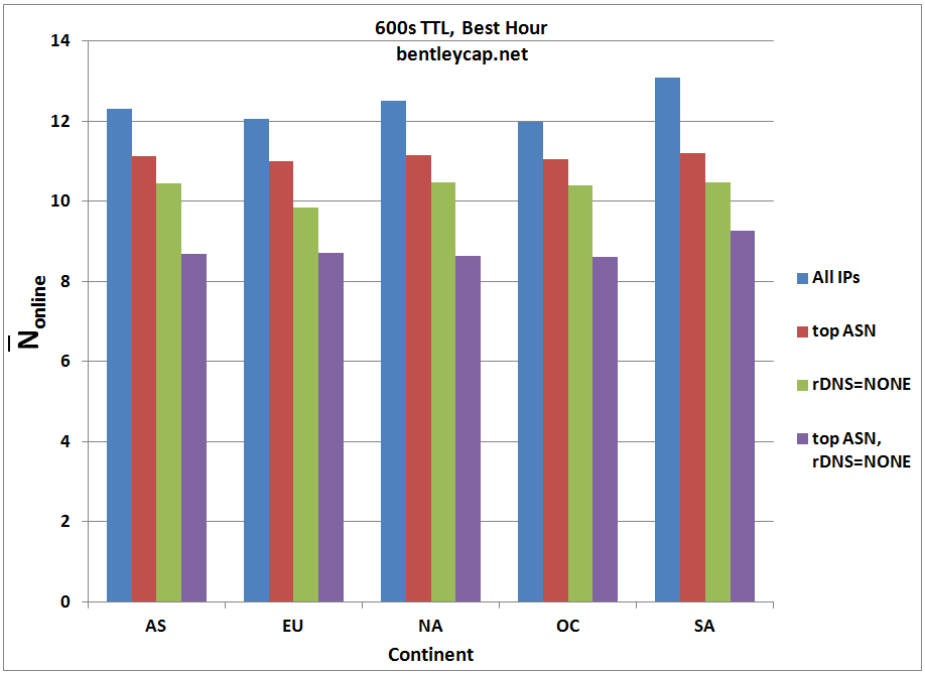


Figure 4.20: Optimal N_{online} for example domain *bentleycap.net* with 600-second T_{ttl}

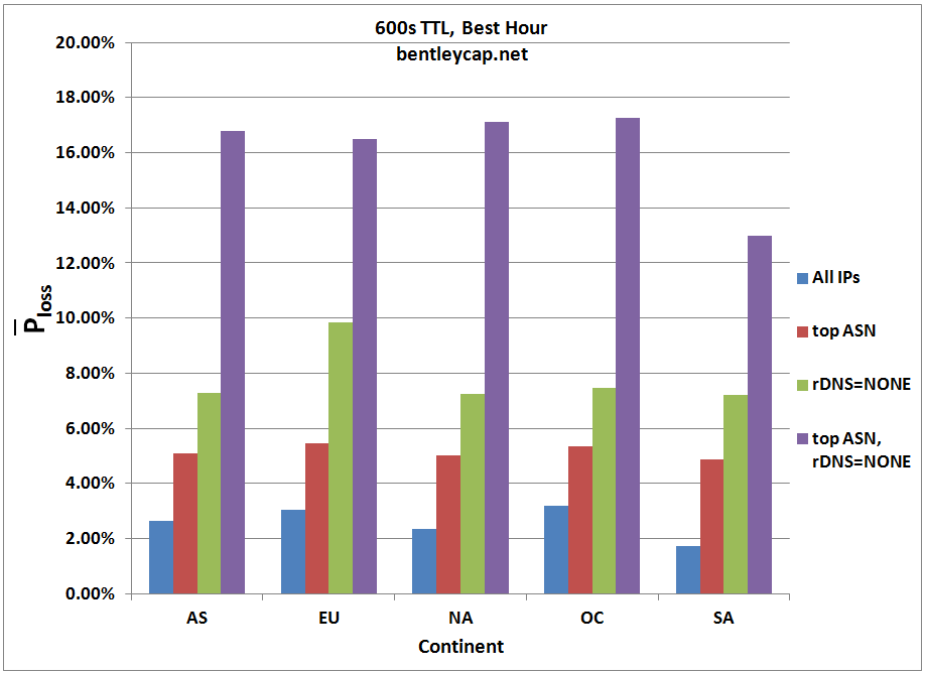


Figure 4.21: Optimal P_{loss} for example domain *bentleycap.net* with 600-second T_{ttl}

4.7 Connectivity Detectors

4.7.1 Good Guys

Thus far, we have shown that current FF botnet resources are capable of supporting mimicry attacks against DNS-based FF detection systems at the expense of online availability and performance. While we have been able to reduce their online availability and performance with additional constraints and detection metrics, we have failed to demonstrate with our models that such an approach can incapacitate botnets, reducing their online availability to zero. Inevitably, current botnet resources provide some mechanism for evading detection, while still preserving some fraction of online, reachable bots. In light of this discovery, we propose a novel detection metric to use in conjunction with existing and future DNS-based FF detection systems: the domain's *percent connectivity*. Percent connectivity measures how many of the advertised IPs are actually online (i.e., have connectivity to the Internet). For benign CDN domains, this ratio is typically low, with the majority of the advertised IPs being online. The unreliable connectivity of botnets, on the other hand, ensures they will intrinsically possess fewer online IPs. Furthermore, we have shown throughout this chapter that botnets can only defeat these DNS-based FF detection systems at the expense of their online connectivity. While successful mimicry attacks against the previous detection systems do not reduce the botnets' online IPs to zero, they may be able to reduce them past a threshold not seen for valid domains, leading to their detection.

We propose a modification to our previously proposed spatial detector to incorporate connectivity as a detection metric. Over every 2 queries, each monitoring node will also calculate the domain's connectivity as the ratio of online IPs to total IPs. The individual node's connectivity results are then averaged to find the domain's *average percent connectivity*. To explore how well this serves as a detection metric

against current FF botnet advertising strategies, we calculated the average percent connectivity for our manually identified FF and CDN domains. As in Section 4.6.1, we make use of DIGGER's extensive 4-month DNS data. For the active duration of each domain, each node calculates the connectivity between every pair of consecutive queries observed. Then, we pair each node's reported connectivity results with those of 4 other nodes, one from each continent, observed in close temporal proximity, calculating the average. The resulting distribution is shown in Fig. 4.22 as a CDF. Unlike the percent-wrong-continent feature, the CDN and FF plots exhibit similar concave trends. From the figure, we see that ≈99% of CDN queries have an average percent connectivity of 80% or greater, compared to only ≈75% of FF domains. It appears that an average-percent-connectivity threshold, O_{thresh}, of 80% would seldom misclassify a CDN domain. Next, we will explore if current botnet resources can both successfully perform mimicry attacks *and* satisfy the new detection constraint, $O_{thresh} = 80\%$.

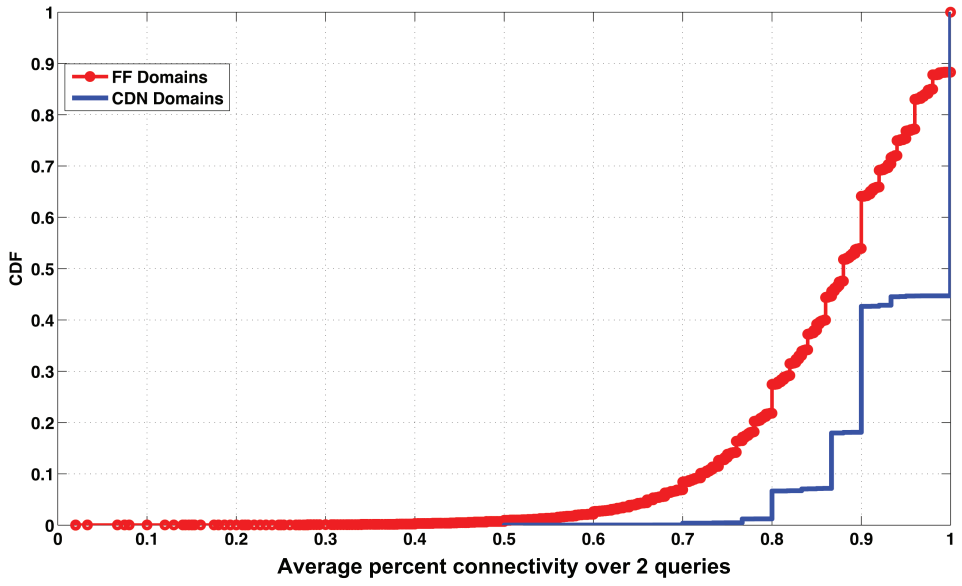


Figure 4.22: CDF of average percent connectivity over 2 queries for FF and CDN domains

4.7.2 Bot Guise

4.7.2.1 Fast-Connectivity Detection Systems

First, we will examine the feasibility of current FF botnets to perform mimicry attack against 2-query, fast-detection systems augmented with a connectivity threshold, which we will term *fast-connectivity detection systems*. We will use a percent-connectivity threshold of $O_{thresh} = 80\%$: *over 2 DNS queries, any domain with an average percent connectivity less than 80% will be flagged as malicious.* As before, we will invoke an N_{thresh}=20, limiting the total IPs seen over any 2 queries to 20. Likewise, we assume botmasters attempt to advertise as many IPs as possible so long as N_{thresh} is not violated, as this maximizes online availability and performance. Once again, we find an optimal $N_{overlap}$ such that $2 * N - N_{overlap} = N_{thresh}$, optimizing Eqs. (4.5) and (4.6). We do this for various T_{ttl} values ranging from 10 minutes to 1 second, shown in Fig. 4.23. As we can see from the figure, the commonly used T_{ttl} values of 300 and 600 seconds cannot successfully perform a mimicry attack against the fast-connectivity detector; current botnet resources could not achieve 80% average connectivity while still defeating the requisite N_{thresh}, resulting in detection. However, we notice that with shorter T_{ttl} values of 120 seconds and less, botnets can defeat both N_{thresh} and O_{thresh}, successfully evading detection. While the addition of O_{thresh} has made the popular T_{ttl} choices of 300 and 600 seconds infeasible for mimicry attacks, more diligent botnet management using shorter T_{ttl} values can still succeed.

4.7.2.2 Spatial-Connectivity Detection Systems

While the addition of average percent connectivity improved upon the fast-detection system, the use of shorter T_{ttl} values within current botnet capabilities still permitted successful mimicry attacks. We will now explore mimicry attacks against our

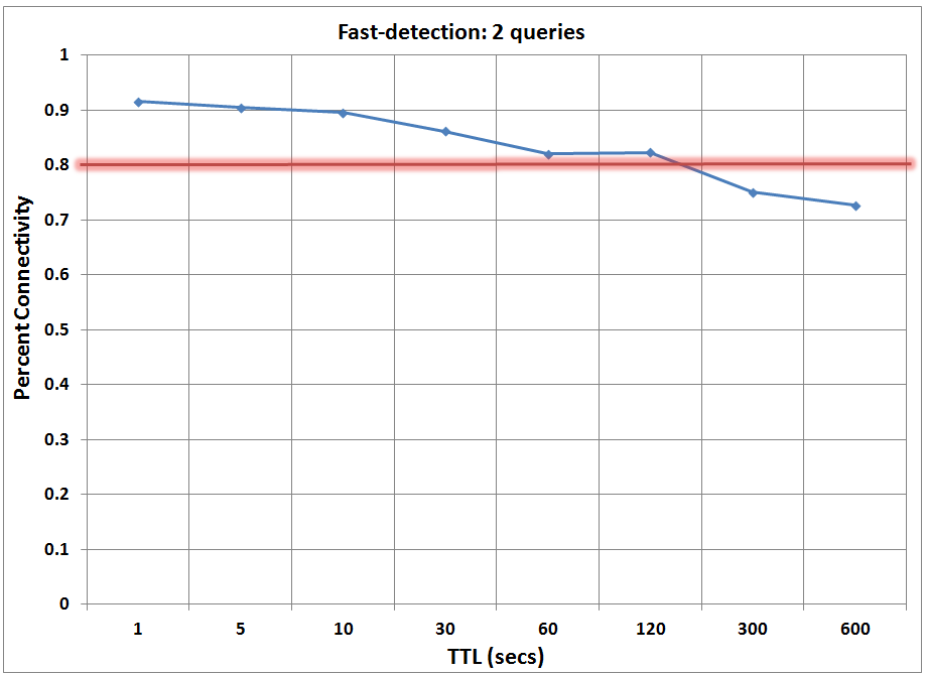


Figure 4.23: Average percent connectivity using 2-query fast detection for various T_{ttl} values

spatial detector enhanced with average percent connectivity, which we term *spatial-connectivity detection systems*. We model the spatial simulator as before in Section 4.6.2, applying the same ASN/rDNS constraints to the continental online-IP model. In addition, we will add constraints allowing the top two ASNs (2nd ASN) and an additional rDNS return result (2nd rDNS). Since these constraints are applied to each continent's DNS-advertisement strategy, the DNS-query results can can contain 3 ASNs, including those due to IPs from other continents, before being flagged as malicious. As before, most FF domains do not contain the requisite resources to support a successful spatial-mimicry attack; most would only be able to provide one or two fresh queries per hour, and when accounting for online-decay, they will easily fall below the average-percent-connectivity threshold. However, as before, we will assume that if they possess the necessary resources to support 2 queries, then they will have those resources during that entire hour. If botnets are still not able to attain the required average percent connectivity under this favorable assumption,

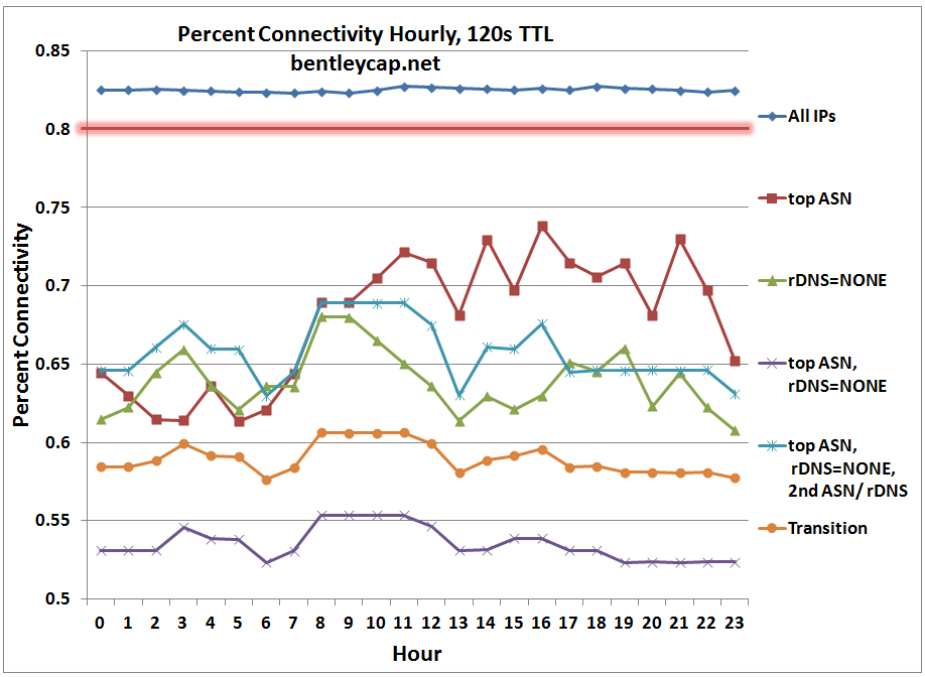


Figure 4.24: Average percent connectivity hourly for example FF domain *bentleycap.net* using 120-second T_{ttl} and 2-query spatial detection for various ASN/rDNS constraints

then such a spatial-connectivity detector holds promise as a fast and viable detection mechanism.

As in Section 4.6.2, we find the optimal advertisement strategy for each hour under the various ASN/rDNS constraints using different T_{ttl} values. Clearly, allowing botmasters to use IPs from a 2nd ASN on each continent increases the amount of online bots available each hour, making it easier to obtain the required percent wrong continent threshold, C_{thresh}. However, the amount of IPs from the 2nd ASN is typically much fewer than from the top ASN, requiring botmasters to substitute new IPs from other ASNs using the ASN-mimicry attack described in Section 4.4.2.1. Consequently, there will exist a transition period where one DNS query must contain only IPs from a single ASN on that continent, while the other will contain IPs from 2 ASNs on that continent. The reduction of ASNs in this transitional query will often require the use of fake IPs from that continent and ASN in order to continue

satisfying C_{thresh}. We can find the average percent connectivity during this transitional period by using the optimal N_{online} under each ASN/rDNS constraint, one for each constituent query, which we term the *transition* constraint in future plots. In Fig. 4.24, we have plotted the average percent connectivity attained hourly under the various constraints for the example domain *bentleycap.net* using a 120-second T_{ttl}. This plot demonstrates how, when under no constraints, the FF domain has sufficient resources to successfully defeat the spatial detector and achieve the necessary threshold for average percent connectivity. However, after applying even a single constraint, the botnet can no longer achieve the required 80% average connectivity, resulting in detection. From the plot, we observe that the constraint with only a single ASN per continent performs worse than the constraint allowing two, with the transition constraint's results falling in between. Within the context of our assumptions thus far, the transitional results best exemplify how we would expect botnets with these resources to perform periodically throughout the hour. While they may often perform better, they will invariably have to transition ASNs throughout the hour, resulting in the transition constraint's performance; if they cannot defeat the detector when transitioning, they will be caught and the mimicry attack considered a failure. We plot the hourly average percent connectivity under the transition constraint, using various T_{ttl} values ranging from 10 minutes to 1 second, for the example domains in Figs. 4.25, 4.26 and 4.27. We find that the smaller domain, *bentleycap.net*, cannot achieve the required 80% connectivity threshold over the entire 24-hour period, regardless of its use of T_{ttl}; even under our favorable assumptions, the botnet does not have enough online resources adequately distributed globally to overcome the spatial-connectivity detector. The moderately sized domain, *www.aquauna-ttour.ru*, also has difficulty with the mimicry attack. For most T_{ttl} values, the botnet cannot defeat the detector at any point during the day. However, with very short T_{ttl} values of 1 and 5 seconds, the botnet can evade detection at 11 a.m. GMT; with a 1-second

T_{ttl}, it can also evade detection at 12 a.m. and 1 a.m. GMT. While the larger botnet, *purgand.com*, cannot defeat the detector at any point with T_{ttl} values of 5 minutes or more, it can occasionally evade detection with 1- and 2-minute T_{ttl} values. Unfortunately, T_{ttl} values of 30 seconds and less allow it to attain an average percent connectivity above the threshold.

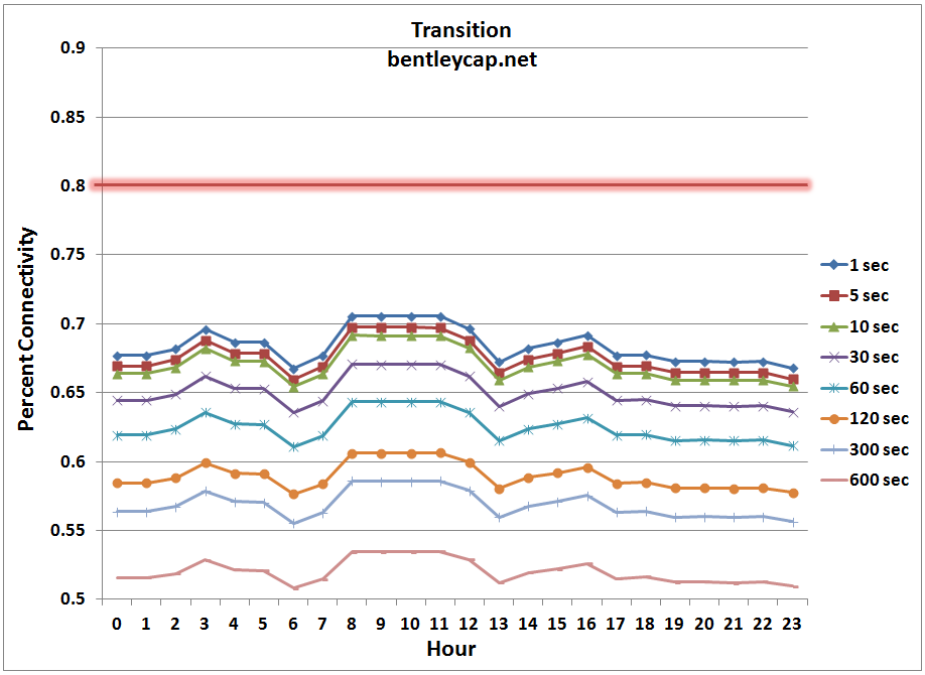


Figure 4.25: Average percent connectivity hourly for *bentleycap.net* under the transition constraint and using 2-query spatial detection for various T_{ttl} values

As we have previously mentioned, we make the assumption that if the botnet's resources can satisfy 2 DNS queries, it can do so for the entire hour, regardless of the T_{ttl} value used. If under these favorable assumptions, botnets using the minimal 1-second T_{ttl} cannot defeat the detector, then the spatial-connectivity detector should serve to successfully combat mimicry attacks. However, *purgand.com* has demonstrated that its resources under these previous assumptions are capable of defeating the detector using short T_{ttl} values. Therefore, we must better represent the actual T_{ttl} capabilities of the botnets in order to determine if they can indeed obtain the small values required to avoid detection. To keep the complexity of the models down,

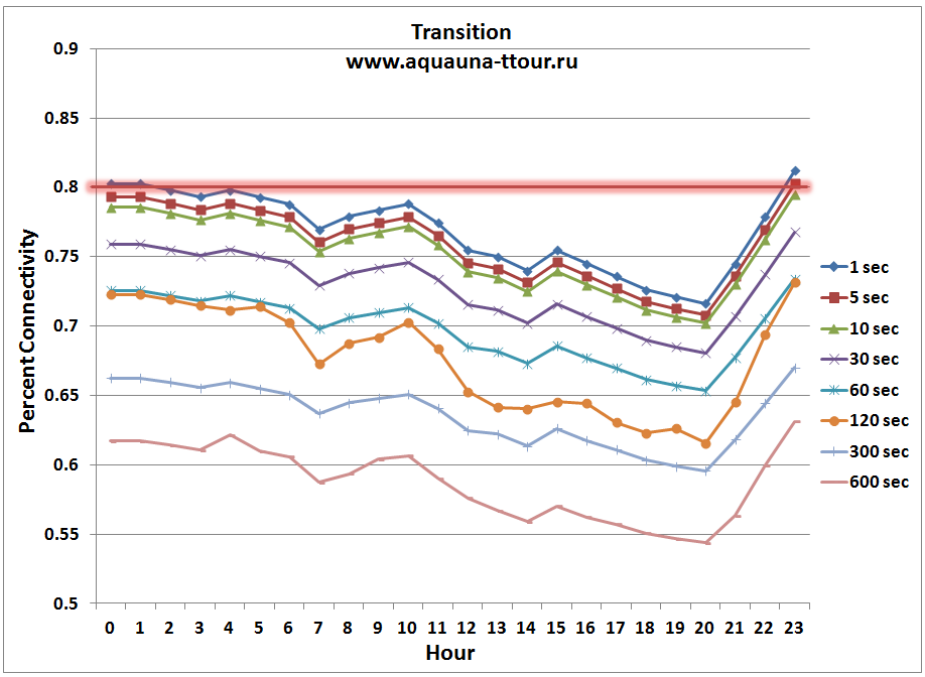


Figure 4.26: Average percent connectivity hourly for *www.aquauna-ttour.ru* under the transition constraint and using 2-query spatial detection for various T_{ttl} values

we will once again adopt the approach of utilizing a favorable, upper-bound assumption, granting botnets their best hope of evading detection. For the representative FF domains, we determine the minimum T_{ttl} value (minTTL) attainable hourly at each continent based on the amount of online bots available to each domain *globally* during each hour. If botnets cannot escape detection under this favorable assumption, then the *actual* resources available at each continent cannot possibly support a successful mimicry attack.

To determine the minTTLs that could be used on each continent hourly, we examine the maximum online IPs available globally under the constraint: top ASN, rDNS=NONE, and 2nd ASN/rDNS. We use the globally available online IPs to determine the minTTLs that could be supported if one new IP were introduced in each query (i.e., $N_{new} = 1$), as this provides the highest online availability for the botnet. Let $N_{global}(h)$ represent the global online IPs the botnet has for an hour, h; then that hour's minTTL is calculated as $TLL_{min}(h) = \lfloor \frac{3600}{N_{global}(h) - N_{overlap}} \rfloor$, where

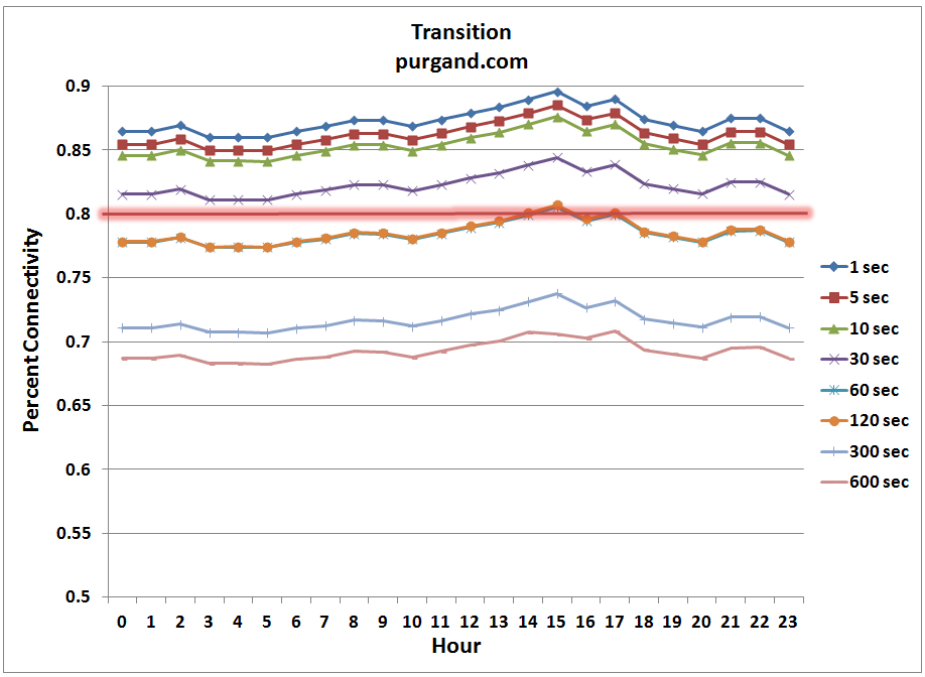


Figure 4.27: Average percent connectivity hourly for *purgand.com* under the transition constraint and using 2-query spatial detection for various T_{ttl} values

$N_{overlap} = 18$ since $N_{thresh} = 20$. The hourly minTTLs for the example domains are shown in Table. 4.5. Next, we find the average percent connectivity attained hourly by the botnets based on each hour's minTTL, shown in Fig. 4.28 for the example domains under the transition constraint. Neither the small (*bentleycap.net*) nor the medium (*www.aquauna-ttour.ru*) botnets can evade detection at any point during the day based on their globally achievable minTTLs. This is unsurprising, as only the medium-sized botnet could evade detection previously, and even then, when using 1- and 5-second T_{ttl} values, it could only do so for a few hours. Interestingly, even under these favorable assumption, the minTTLs possible for the large botnet (*purgand.com*) cannot support the mimicry attack continuously over 24 hours. There are 5 hours during the day—the prime European evening hours of 8 p.m. to 12 a.m. GMT—when the average-percent-connectivity threshold cannot be reached, allowing the botnet to be detected within a 24-hour period. In Fig. 4.29, we plot the minTTLs *purgand.com* can achieve hourly with its globally online bots under no constraints

and under the constraint: top ASN, rDNS=NONE, 2nd ASN/rDNS. Under no constraints, *purgand.com* possesses enough online bots globally to support a minTTL of 3–4 seconds each hour. With no spatial-detection mechanism, this would be sufficient to defeat a fast-connectivity detection system. Even under our favorable assumptions for the spatial-mimicry attack, the large botnet could still avoid detection. However, once applying such constraints on the ASN and rDNS values returned in each continent, the domain no longer has sufficient resources to defeat the detector 24 hours a day. These results imply that a spatial-connectivity detection system could be used to combat FF mimicry attacks, as actual botnet resources will be fewer than our favorable assumptions permit.

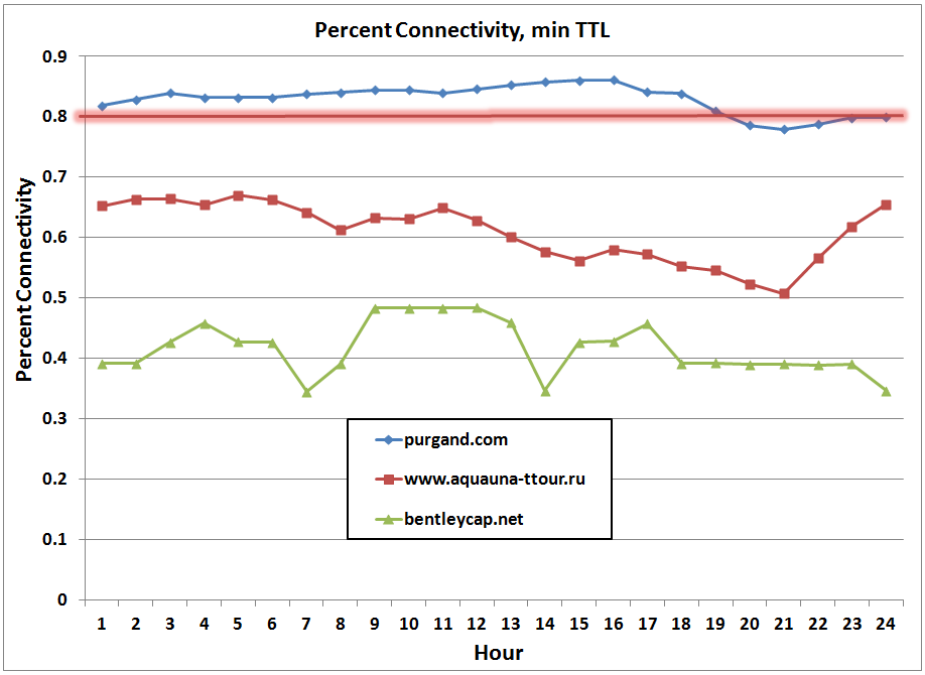


Figure 4.28: Average percent connectivity hourly for example FF domains under the transition constraint and using 2-query spatial detection for each hours' minTTL

hour	Minimum Possible TTL (sec)		
	purgand.com	www.aquauna-ttour.ru	bentleycap.net
0	28	171	900
1	21	150	900
2	17	144	720
3	16	156	600
4	16	133	720
5	16	138	720
6	15	171	1200
7	16	211	900
8	16	180	514
9	16	189	514
10	16	156	514
11	15	180	514
12	14	211	600
13	14	276	1200
14	16	300	720
15	19	276	720
16	25	276	600
17	30	327	900
18	40	327	900
19	56	400	900
20	59	450	900
21	59	300	900
22	50	211	900
23	42	180	1200

Table 4.5: minTTL (seconds) per hour for example FF domains under constraint: top ASN, rDNS=NONE and 2nd ASN/rDNS

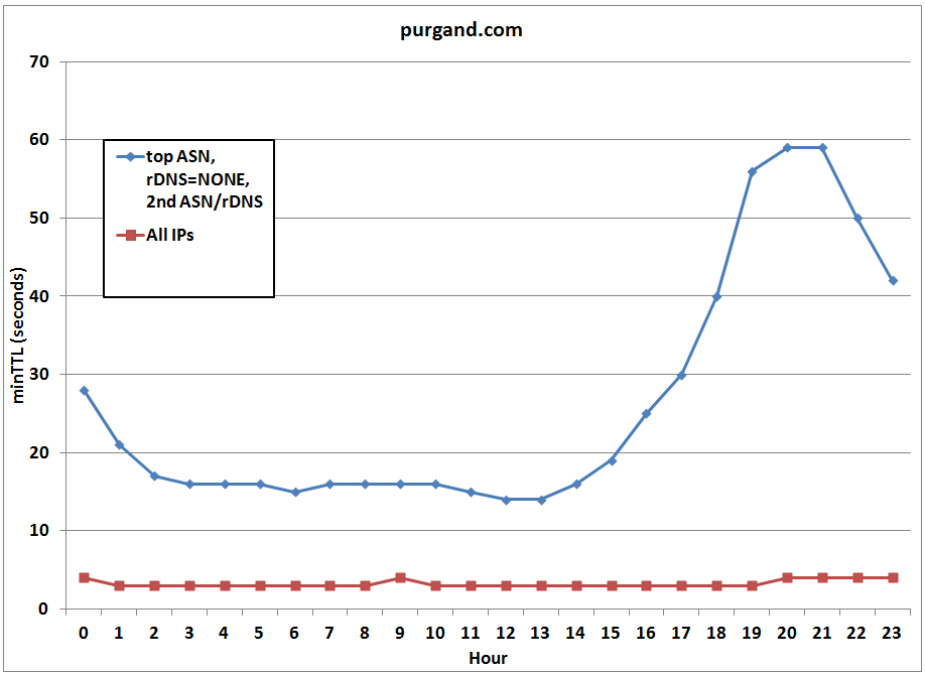


Figure 4.29: Hourly minTTL possible for *purgand.com*, based on globally available online bots, under no constraints and under constraint: top ASN, rDNS=NONE and 2nd ASN/rDNS

4.8 Conclusion

In this chapter, we examined the current, state-of-art, DNS-based FF detectors and analyzed their effectiveness in detection. We evaluate the feasibility of novel mimicry attacks against such systems using accurate models we developed for bot decay, online availability, DNS advertisement, and performance. Combining these models with empirical evidence and logical assumptions, we have shown that current botnet resources are sufficient for subverting state-of-art, DNS-based FF detectors via mimicry attacks. Additionally, we uncovered evidence of current FF domains already adopting aspects of our proposed mimicry attacks, although their observed management is less meticulous than our optimal models assume. Nevertheless, as detection systems improve and become more pervasive, we expect botmasters will increase their diligence in IP management to extract the most from their botnets' resources. We have found that incorporating more advanced views, such as an extended detection window, can significantly encumber mimicry attacks through the introduction of additional, necessary parameters unknown a priori to botmasters. However, since security by obscurity is always a bad idea, ultimately, they are still susceptible to determined adversaries; for example, trial-and-error reconnaissance missions could derive the detection window size and type. Still, subverting the more advanced detectors requires significantly more effort from botmasters, making a strong argument for their adoption in augmenting the simpler, fast-detection systems. We have proposed a novel spatial-detection system utilizing 5 coordinating DNS-monitoring nodes on different continents. Even without the necessary global distribution of online bots to perform location-aware DNS-advertisement strategies, through the use of fake IPs, botnets can defeat spatial detection at the expense of online availability and performance. The observation that current botnet resources continue to defeat DNS-based FF detection systems at the expense of online availability led us to introduce another novel detection metric, percent connectivity. We have shown that the incorporation

of percent connectivity can significantly improve fast-detection systems, requiring increased diligence by botmasters to evade detection using short T_{ttl} values. Lastly, we have shown that current FF botnets do not posses the online resources required to defeat our spatial-connectivity detection system. The additional spatial constraints, combined with the ASN/rDNS constraints, overly strains botnet resources, with even the largest botnets unable to perform mimicry attacks 24 hours a day under favorable assumptions. We hope that showing the mimicry potential currently attainable by FF domains will foster improvements to existing detection systems as well as provide new insight into the adaptive limitations of FF botnets. By demonstrating the effectiveness of spatial-connectivity detectors against mimicry attacks—and percent connectivity as a supplemental feature in any DNS-based FF detection system—we hope to encourage the development and improvement of such techniques in future DNS-based FF detection systems.

CHAPTER V

Can Open WiFi Networks Be Lethal Weapons for Botnets?

5.1 Introduction

Currently, much research has been done to understand and detect existing botnets as well as predict their future capabilities and C&C channels. However, at this time, little research has been done examining mobile botnets, i.e., botnets composed entirely of compromised mobile devices. To be fair, this is primarily due to the lack of mobile botnets in the wild; at present, mobile botnets do not enjoy the same popularity as their more traditional brethren. Nevertheless, with their booming application markets, standardized OSes and rapid advances in processing power and memory, mobile devices are capable of increasingly sophisticated tasks, approaching those of modern computers. Combined with their multiple communication interfaces (i.e., WiFi, 3G/4G, Bluetooth, SMS and MMS messaging) and always-on connectivity, they are capable of sophisticated attacks not possible with traditional computers. For example, a mobile botnet could potentially cripple local cellular communication by flooding the network's service towers with calls or SMS/MMS messages. As a result, mobile devices are quickly becoming an attractive target for botmasters, and it is only a matter of time before mobile botnets emerge on the Internet threatscape.

In this chapter, we evaluate the potential for mobile botnets to communicate and perform nefarious actions solely over open WiFi Access Points (APs), which we term *mobile WiFi botnets*; to the best of our knowledge, we are the first to research this scenario. The use of open WiFi networks for mobile botnets can provide a higher level of stealthiness and has fewer barriers to entry than other communication mediums, which we discuss further in Section 5.3.1. In assessing the feasibility of mobile WiFi botnets to support botnet C&C, DDoS attacks and spam attacks, we make the following contributions. First, we design a proof-of-concept mobile WiFi botnet, including its C&C, DDoS attack and spam attack protocols as well as multiple AP-selection algorithms designed to exploit the predictable mobility patterns of public transportation. Second, we build a mobile WiFi botnet simulator, using accurate timing models for AP association, Internet communication and achievable wireless throughput based on the mobile bot's distance from the open AP. Third, we run this simulator for various attack scenarios using real-world cab mobility traces, bus routes and actual open WiFi AP locations for the urban environment of San Francisco.[1] Fourth, through in-depth simulations, we demonstrate that mobile WiFi botnets can support rapid command propagation,[2] can successfully mount DDoS[3] and spam[4] attacks, and are sufficiently distributed across open WiFi networks—with no single network being over-utilized at any given moment—to make detection difficult. Moreover, those bots able to receive commands usually have ≈30–50% probability of being

[1]We enhance the cab traces and bus routes to provide accurate location information at a one-second granularity.

[2]For cabs, commands typically reach more than 75% of the botnet within 2 hours of injection—sometimes, within as little as 30 minutes after injection. For buses, more than 80% of the botnet is often reached in under an hour.

[3]The bus botnet was typically capable of issuing over 2 million SYN packets per hour (≈555 per sec) and as many as 4 million (≈1,100 per sec) during its peak hour. The smaller cab botnet was typically capable of issuing 0.8–1.2 million SYN packets per hour (≈222–239 per second) and as many as 1.7 million (≈472 per sec) during its peak hour.

[4]The bus botnet could issue between ≈600,000 and 11.7 million spam emails daily and over 1 million per hour in certain instances. During weekday rush hours (i.e., 8 hours), the cab botnet could issue between ≈150,000 and 2.3 million spam emails daily. On weekends, it could issue between ≈400,000 and 6.4 million spam emails daily.

able to do so within a minute of the command being issued. Finally, we present and evaluate some defensive strategies against future mobile WiFi botnets.

The remainder of this paper is organized as follows. Section 5.2 gives an overview of related work. Section 5.3 describes the type of mobile WiFi botnet we examine, explaining how mobile devices could become infected, our threat model, and how C&C, DDoS attacks and spam attacks can operate over only open and unencrypted WiFi networks. Section 5.4 describes the datasets used in our experiments, the AP-selection algorithms we examine and a detailed breakdown of the C&C, DDoS and spam attacks used in our simulations. Section 5.5 analyzes our simulation results, while Section 5.6 describes and evaluates some potential defenses against mobile WiFi botnets. Finally, Section 5.7 offers our concluding remarks.

5.2 Related Work

Despite the rich literature on botnets in the Internet environment, relatively little has been done to study and understand the same threats in mobile environments. In recent years, there has been a surge of mobile malware which have already started to demonstrate botnet-like traits. For instance, SymOS.Yxes [68], discovered in early 2009, reports user-sensitive information back to a centralized server through an HTTP-based C&C protocol. `Ikee.B` [46], targeting jailbroken iPhones, uses a similar HTTP-based mechanism to connect to a control server, download additional components and steal user information. There have also been several research efforts to design advanced C&C protocols for mobile botnets. Singh *et al.* [64] evaluated the feasibility of using Bluetooth as a medium for botnet C&C; they demonstrated that, due to the repetitive nature of human daily routines, such a Bluetooth-based C&C infrastructure allows the propagation of C&C messages to ≈67% of infected devices within a day. Muliner [52] proposed SMS and SMS-HTTP hybrid C&C protocols to facilitate the communication between compromised smart phones. Xiang *et al.* [74]

introduced the design of Andbot for Android phones, which exploits URL Flux as a stealthy and robust C&C channel. Taking a different angle, Traynor *et al.* [61] demonstrated the impact of DDoS attacks against the core of cellular networks utilizing compromised mobile phones. Their idea was to issue resource-consuming service requests to overwhelm the Home Location Register (HLR), a critical component providing users' location information. Their focus is on the characterization of large-scale attacks, whereas our work investigates the effectiveness of using open WiFi networks as a stealthy channel to coordinate a large number of moving bots and launch DDoS and spam attacks. Lastly, Akritidis *et al.* [17] explore how attacks can spread from stationary host to stationary host using urban WiFi networks. While it may be possible to construct a botnet using their techniques, this possibility is only briefly mentioned in their work. In this chapter, we focus on the nature such a botnet if it were to exist and function solely over open and unencrypted WiFi networks, studying the complexities of mobility and its effect on botnet attacks and stealthiness.

5.3 WiFi-Based Mobile Botnets

In this section, we first explain the advantages open WiFi networks provide to mobile botnets over other communication interfaces. Next, we describe how a mobile botnet can use open WiFi networks for C&C, DDoS attacks and spam attacks, followed by our threat model and assumptions.

5.3.1 Why WiFi

One of the challenges facing mobile botnets is the omniscient view cellular providers have of their networks, making it easier to discern coordinated botnet activities. All cellular network communication—including SMS/MMS messages and 3G/4G data—can be observed by the cellular provider, and all devices participating on the network must first authenticate using a non-spoofable mobile ID. Therefore, mobile bots uti-

lizing only the cellular network for botnet activity can be more easily detected and neutralized. Moreover, since mobile IDs can't be spoofed, infected devices can be denied network access until the bots have been removed. Consequently, the use of a cellular network for botnet activities can result in the compromised devices being quickly detected and removed from the botnet.

To be effective, mobile botnets will have to adopt novel, stealthy approaches for their communication and attacks, exploiting some of the other interfaces available to mobile devices. It has been shown by Singh *et al.* [64], that Bluetooth can operate fairly successfully as a covert C&C channel, with ≈67% of their proof-of-concept botnet receiving commands within 24 hours of injection. Unfortunately, while stealthy, a Bluetooth C&C channel suffers from a significant barrier to entry: to successfully transmit commands, a large number of identically infected, slowly moving devices must continuously come in close proximity to one another. This constraint—due to Bluetooth's limited communication range (typically only a few meters) and its slow transmission rates (typically less than 1 Mbit/s)—makes Bluetooth C&C unsuitable for newly emerging mobile botnets, where the infected devices may be few and geographically dispersed.

Cellular-network interfaces, such as SMS/MMS and 3G/4G, can provide a more feasible communication channel for small, newly emerging botnets than Bluetooth. However, their inherent lack of stealth can result in rapid detection and mitigation, ultimately rendering the botnet impotent. On the other hand, a Bluetooth-based mobile botnet, though stealthy, can only function once the amount of infected devices in a given area has reached a certain critical mass and density. The use of WiFi networks for botnet C&C and attacks resolves both issues. Unlike Bluetooth, a fledgling mobile WiFi botnet's dispersed bots can communicate and participate with other bots around the world, transmitting at rates an order-of-magnitude greater than Bluetooth. Furthermore, the use of WiFi networks for C&C and attacks de-

prives cellular providers of their complete network view, hindering detection. The use of WiFi grants additional stealthiness to botnets since nearly all WiFi APs use Network Address Translation (NAT) to support multiple, simultaneously connected wireless devices—all sharing a single external IP address for Internet access. As a result, botnet activity is hidden amidst a plethora of benign traffic generated by the WiFi AP's other users, making detection by the AP's Internet Service Provider (ISP) increasingly difficult. Additionally, ISPs cannot simply deny service to a mobile WiFi bot's IP address, since this results in *all* the WiFi AP's connected devices—most of which are benign—also losing service. Detection can be further inhibited by spreading botnet activity across many *different* WiFi networks, limiting the amount observable at any given network. In fact, the less botnet activity performed behind a single WiFi AP, the more difficult detection becomes. Mobile bots are well suited to take advantage of this technique, as their inherent mobility can result in exposure to multiple WiFi APs throughout the day. Despite the persistent connectivity available when at the home or office, mobile bots have minimal opportunities for spreading their nefarious traffic across multiple networks, limiting their potential stealthiness. However, when commuting, mobile bots come in contact with numerous WiFi networks and can discretely distribute their activity across any number of them. Even when they are detected, the bots' mobility ensures they will soon have access to other WiFi networks. By spoofing IPs and leveraging users' mobility to obtain short online sessions over many different WiFi networks, mobile WiFi botnets could grant botmasters an unprecedented level of stealth.

As a result, in this chapter, we examine the effectiveness of mobile WiFi botnets during both weekdays and weekends. For our cab botnet, our weekday analysis is limited to morning and afternoon office-commute hours, though this time period is not a requirement. Clearly, mobile bots will make use of open WiFi networks whenever they are within range, regardless of the time of day. However, during the weekdays,

office-commute hours are the most likely time when bots will be moving through multiple networks, allowing communication to be at its stealthiest and impeding detection. Moreover, we will examine mobile bots traveling in vehicles, as their rapid pace ensures—with the exception of traffic jams, lights and refueling—that a bot will be within range of a given WiFi AP for typically just a few seconds. If the limited duration of quickly moving vehicles is sufficient to perform botnet C&C and DDoS and spam attacks, then the more leisurely pace of bikers and pedestrians will have no difficulty doing so as well.

Moving through an urban environment in a vehicle, a mobile bot will have access to numerous home and business WiFi APs. While the vehicle's speed and movement ensures that the mobile bot will continuously have access to new WiFi networks, it also ensures that the amount of time on any given network is limited. Therefore, it isn't practical for a mobile WiFi bot to attempt hacking into an encrypted or closed network for online access. Rather, by only utilizing open and unencrypted WiFi networks, mobile bots can quickly perform a small subset of malicious activities on each network before moving out of range. Despite the tightening of home WiFi network security, necessity dictates that many businesses (e.g., restaurants and cafes) leave their WiFi networks open and unencrypted. Inconveniencing customers with a password before granting network access could potentially drive them to competitors with open networks, especially if they have to wait in line to obtain the password. While many businesses employ a semi-open network, requiring a customer to click on an "Accept" button before granting access, this does little to defend against mobile WiFi botnets (see Section 5.6). Consequently, while the prevalence of open and unencrypted WiFi networks is dwindling, between businesses placating customers and home users indifferent to security, they will not entirely disappear, making them a useful medium for mobile botnets.

5.3.2 Mobile Botnet

Since most businesses with open WiFi networks limit online activity to web traffic over port 80, a mobile WiFi botnet should use an HTTP-based C&C channel; this provides the added benefit of hiding C&C traffic among other, benign HTTP traffic on the network. Traditionally, bots utilizing an HTTP-based C&C periodically poll the command server for new commands. The command server is typically reached using a deterministic yet changing domain name, making it difficult for defenders to predict and block the malicious domains using such techniques as a DNS sink hole. Requests to the current command server's domain are often serviced by proxy bots, further protecting the botmaster from detection and disruption. While a mobile WiFi bot could still poll periodically throughout the day, this could result in numerous polls originating from a home or office WiFi network (i.e., where the user spends the majority of his time), increasing the probability of detection. A more stealthy approach would be to limit periodic polling to when the bot is connected to an open WiFi network, particularly one that is *not* a home or office network.[5] Depending on the amount of control and stealthiness required by the botmasters, mobile bots can poll once or multiple times per open network. In this way, an HTTP-based C&C can be modified for a mobile WiFi botnet, providing botmasters with an unprecedented level of anonymity and protection. Likewise, DDoS and spam attacks can be issued from multiple open networks as they are in range, issuing small portions of the total attack at each AP. In Section 5.4.4, we will discuss in detail how our prototype mobile WiFi botnet actually achieves HTTP-based C&C and performs DDoS and spam attacks.

[5]Home and office networks can easily be determined based on the duration of connectivity as well as the time of day; mobile devices are likely to spend the evenings on a home network and business hours on an office network.

5.3.3 Threat Model

Our focus in this paper is to determine the effectiveness of open WiFi networks for botnet C&C, DDoS attacks and spam attacks using high-mobility devices (i.e., those traveling in vehicles). We are not concerned with the problem of mobile device infection, which is out of the scope of this paper and has been previously researched. For example, due to homogeneous mobile OSes, malware can spread through downloadable content (e.g., applications and ringtones), infected SMS/MMS messages, email, and browser exploits. Instead, we assume all devices are infected with the same bot malware, allowing us to ignore the complications of infecting heterogeneous devices. It seems obvious that mobile devices will have sufficient online access when connected to a user's home or office WiFi network. However, at this point, little/no research has been done to determine if botnet activities can be supported using *only* open WiFi networks, which can significantly encumber detection. In our model, we assume the existence of a small mobile botnet operating solely over open and unencrypted WiFi networks in a single metropolitan city. The botmaster's adversarial goals are three-fold. The first goal is to obtain a sufficient amount of control over a modest portion of the total botnet during the bots' most transient and high-speed period, which is the most difficult to control. This period occurs when bots are traveling in a vehicle, and we will determine if this goal can be achieved by designing and simulating a C&C protocol for a mobile WiFi botnet. We will explore what amount of the total potential botnet is actually reachable, how frequently bots can receive commands and how quickly new commands propagate throughout the botnet. The second adversarial goal is to use the small mobile WiFi botnet to issue successful DDoS and spam attacks. Likewise, we will design and simulate such attacks for a mobile WiFi botnet to evaluate its feasibility. The final goal is that both botnet C&C and its attacks can function in a stealthy manner, which we will evaluate by ensuring that the malicious botnet's traffic is adequately dispersed over multiple WiFi APs.

In addition to the above goals, we considered whether a botmaster could make use of mobility patterns known *a priori* to improve botnet performance in any of the three goals. We noticed that with knowledge of the exact mobility patterns of the bots, botmasters could make intelligent choices for which APs to connect to and potentially improve their C&C and attacks. This approach is further described in Section 5.4.3

5.4 Experimental Setup

5.4.1 Description of Datasets

We now describe the datasets used in our experiments to determine open WiFi APs and simulate vehicular mobility patterns in an urban environment.

5.4.1.1 Open WiFi APs

To obtain a comprehensive dataset of open WiFi APs, we made use of the Wireless Geographic Logging Engine (WiGLE) dataset for downtown San Francisco [11]. WiGLE contains an extensive database of wireless AP information, built from a collaborative effort of thousands of researchers and WiFi enthusiasts. Using WiFi scanners, individuals report on an area's WiFi topology, including such information as the AP's MAC, geographic coordinates, open or closed status and use of encryption. Removing any APs not unanimously identified both as "open" and unencrypted, our resulting dataset contains the location of 2,349 open and unencrypted WiFi APs in downtown San Francisco, ideal for the purposes of the proposed mobile WiFi botnet.

5.4.1.2 Cab Mobility Traces

To simulate vehicular mobility patterns in an urban environment, we use the mobility traces of taxi cabs in the San Francisco Bay area [2], first published in [58]. The dataset contains mobility traces for 536 cabs over 30 days. Unfortunately, the cabs'

location granularity ranges from seconds to minutes, making it too inconsistent and course-grained for our purposes. To overcome this limitation, we converted the traces into the TIGER [10] map coordinate system, removing any resulting from misbehaving GPS devices. Combining this with the VanetMobiSim [38] vehicular simulation system, we simulated the path and speed between any two cab-trace points based on San Francisco's actual road topology. The resulting dataset contains detailed location information at a one-second granularity for 536 cabs over 24 days in downtown San Francisco. This rich, finely grained mobility dataset will allow us to represent the movement patterns of vehicles during weekday office commutes and weekends.

5.4.1.3 Bus Mobility

To simulate public transportation patterns in an urban environment, we use the San Francisco Bay area's bus route information made available by [9]. The dataset contains detailed bus and trolley route information—broken down into their constituent trips—for weekday, Saturday and Sunday services, indicating each stop location, time and duration. Unfortunately, only knowing the location of buses/trolleys when they arrive at their stops is too coarse-grained for our purposes. To overcome this limitation, we converted the route information into the TIGER [10] map coordinate system and combined it with the VanetMobiSim [38] vehicular simulation system, simulating the path and speed between any two bus stops based on San Francisco's actual road topology. The resulting dataset contains detailed location information at a one-second granularity, allowing us to accurately represent the regular movement patterns of public transportation in downtown San Francisco.

5.4.2 Modeling Open WiFi APs

With the requisite information unavailable, we simplify our simulation environment by assuming each AP has the same signal strength, ignoring attenuation due

to competing wireless signals and environmental obstructions.[6] Then, by fitting exponential curves to the empirical data in [12]—which gives achievable throughput based on distance for 802.11b/g routers—we find Eq. (5.1), allowing us to calculate a mobile bot's average throughput (Mbit/s) based on its current distance in meters, d, from an open WiFi AP. Using our improved mobility dataset (Section 5.4.1), one of our various AP-selection algorithms (see Section 5.4.3) and Eq. (5.1), our simulator calculates each mobile bot's throughput at a one-second granularity, which we use to determine if open WiFi networks are suitable for supporting botnet activities.

$$f(d) = \begin{cases} 37.5 & :_{d \leq 1.52} \\ 37.97 * e^{-8.15*10^{-3}*d} + 4.154 * 10^{-4} * e^{0.33*d} & :_{1.52 < d \leq 22.86} \\ 3379 * e^{-0.22*d} + 22.48 * e^{-0.042*d} & :_{22.86 < d \leq 53.34} \\ 0 & :_{d > 53.34} \end{cases} \tag{5.1}$$

5.4.3 AP-Selection Algorithms

This section discusses the various AP-selection algorithms we examine in this chapter for bus botnet simulations. Since the cab botnet simulations represent individual commutes, which are more difficult to predict than public transportation routes, it only makes use of the Naïve protocol below. Bus routes are more amenable to AP-selection optimization. The routes are public knowledge, and once a bus leaves a stop, bots can check nearby WiFi APs to determine the current route and then optimize AP selection. Therefore, the bus botnet simulations will explore how all of the following AP-selection algorithms affect botnet activities and attacks.

Naïve utilizes 802.11's simple AP-selection algorithm. When a mobile bot is within range of open WiFi networks, it chooses the network with the strongest signal strength, performing botnet activities until it is out of range and must connect to

[6] It is trivial for us to relax this assumption if and when the required information becomes available.

a new AP. Such a naïve AP-selection approach is easily implemented by botmasters and works equally well in any urban environment, requiring no *a priori* knowledge. If a mobile WiFi botnet can operate under this simple AP-selection algorithm, more complicated mechanisms can potentially achieve better results.

Setcover takes advantage of the predictable nature of bus routes and attempts to minimize the amount of expensive AP reassociations performed by bots (i.e., AP switches). Using the offline bus route and open WiFi AP data from Section 5.4.1, we can determine which APs are within range of each bus at a one-second granularity. Finding the minimal number of AP switches then becomes a set cover problem in graph theory, treating each second as a node and each continuous timespan when a given AP is within range as a set. Thus, we want to select the least number of sets (i.e., AP switches) such that the bots will always be connected to an AP if one is available. Unfortunately, set cover is an NP-complete problem, so we approximated the solution with a greedy algorithm: until all seconds with an AP in range are covered, we iteratively choose a set covering the most uncovered seconds.

Weighted Setcover is a modified version of Setcover that takes the bots' distance from the AP into account when choosing sets. This can prove an influential factor in an attack's success because the transmission rate decreases with distance, as shown in Eq. 5.1. Therefore, Weighted Setcover weights each set by the ratio of the uncovered seconds it contains to its average distance. For example, if two APs cover the same number of seconds, the one closer, on average, to the bus will have a larger weight and thus be selected. This minimizes AP switches while maximizing data throughput, potentially resulting in more successful attacks.

Patched is a simple heuristic applied to the other AP-selection algorithms for the bus botnet simulations that takes advantage of the idle time at bus stops; if long enough, a bot could benefit by switching to a closer AP for an increased data rate. At each bus stop, we perform a three-fold test: if 1) the currently selected AP is at

a distance greater than a threshold of 22.86 meters,[7] 2) there is another AP closer than 22.86 meters, and 3) the idle time at the bus stop is sufficient to restore the attack to its state prior to switching APs, then the bot switches to the nearest AP. This attempts to further improve attacks by increasing the data rate when bots are stationary for long periods. Interestingly, we discovered that Weighted Setcover and its patched version produced virtually identical results, indicating that the inclusion of distance in Weighted Setcover naturally takes advantage of the increased throughput available from nearby APs when bots are idle at bus stops.

5.4.4 Attack Protocols

5.4.4.1 C&C Protocol

Fig. 5.1 gives a high-level representation of our HTTP-based C&C protocol between a mobile bot and an open WiFi AP. To acquire commands, mobile bots must first connect to an open WiFi AP (STAGE 1). Once connected, bots must query for new commands from the botnet's command server. This is accomplished by first issuing, over UDP, a DNS query (STAGE 2) on the command server's domain name, likely chosen from a set of predetermined candidate domains. Upon receiving a DNS response, bots must establish a TCP connection (STAGE 3) with the command server. Then, bots can issue an HTTP GET (STAGE 4), requesting the current commands. When the complete HTTP response is received—as indicated by the closing FIN packet—the mobile bot has successfully received the current commands and can begin acting on them. Notice that the mobile bot only successfully receives commands if it remains connected to a given open WiFi network long enough to complete all four stages. Premature interruption of the protocol due to loss of connectivity (likely due to the bot's mobility), resets the protocol to STAGE 1 with the nearest open AP. In our simulations, after successfully completing STAGE 4, our mobile bots immediately

[7]Based on Eq. (5.1), the data rate drops considerably for distances greater than 22.86 meters.

attempt to receive another fresh command by returning to STAGE 2; this process continues, so long as they remain connected to the same open AP, allowing us to determine the finest level of control available.

To determine the timing for STAGE 1, we turn to [24], which reports an average of 2.757 seconds for a mobile device (traveling in a vehicle) to scan for an open WiFi AP, associate with it and obtain an IP. For the later stages, we must account for 802.11b/g's overhead and transmission rates as well as wired communications between the open WiFi AP and Internet servers (i.e., DNS and botnet command servers). In 802.11b/g, a 24-byte PLCP preamble and header must be transmitted at a constant 1 Mbit/s before any subsequent transmission; this constant, 0.192-millisecond overhead is incurred for every wireless transmission in STAGES 2, 3 and 4. Messages are then transmitted at a rate determined by the bot's distance from the WiFi AP and Eq. (5.1), recalculating the rate every second. To simplify the complexity of Internet communications, we use the median Round Trip Time (RTT) of 0.086 second for cable connections in the USA [20] to estimate the RTT between the WiFi AP and Internet servers. The packet sizes of TCP's 3-way handshake in STAGE 3 and the FIN message in STAGE 4 are well defined. For the DNS request and response in STAGE 2, we choose a message size of 100 bytes, which provides ample space for complicated domain names in the DNS request and multiple server IPs in the response. For the HTTP messages in STAGE 4, we choose a message size of 512 bytes, as this will remain within a single packet, is close to the average Internet's packet size and will support complicated GET requests (e.g., reporting device capabilities) and multiple, complex commands in the response.

5.4.4.2 DDoS Protocol

Botmasters achieve the required coordination for DDoS attacks via the C&C channel, informing bots of future attack targets and times. Later, at the appropriate times,

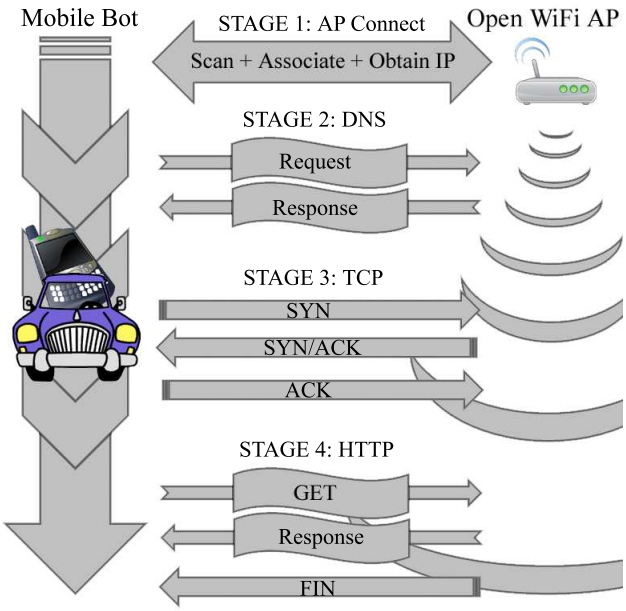


Figure 5.1: Mobile bot's C&C protocol

mobile bots with access to open WiFi networks perform the synchronized DDoS attacks. Attacks are hidden among benign traffic and dispersed across multiple open WiFi networks, making mitigation difficult.

Unlike the C&C protocol in Fig. 5.1, mobile bots only require an online connection (STAGE 1) before they can begin a DDoS attack. As they move, bots continue to connect to different open WiFi APs, issuing a stream of SYN packets at the target for the duration of the attack. Since we know the average time for STAGE 1 and the size of SYN packets, we use Eq. (5.1) and the bots' locations to simulate a DDoS attack originating from a mobile WiFi botnet and evaluate its effectiveness.

5.4.4.3 Spam Protocol

As with the DDoS attack, botmasters achieve the required coordination and setup for spam attacks via the C&C channel, informing bots of the attack time, target emails and the spam message body. Later, at the appropriate times, mobile bots with access to open WiFi networks perform the spam attacks. Unlike the C&C and

DDoS attacks, the amount of transmission overhead in a spam attack is much more nebulous. Therefore, we first examine its lower bound, where each email recipient resides at a different domain. In this scenario, which we term *spam_lower*, mobile bots connect to an open AP, locate the mail server using DNS and then establish a TCP connection, as in STAGES 1, 2 and 3 of Fig. 5.1. Next, they communicate with the mail server using SMTP as shown in STAGES 4 through 9 of Fig. 5.2. Upon completing STAGE 9 and successfully issuing a spam email, bots return to STAGE 2, querying DNS to locate the next target's mail server and continue the attack.

If botmasters don't require customized spam messages for each target and multiple targets reside at a single mail server, transmission overhead could be reduced by issuing to multiple email recipients at STAGE 6. For attacks such as these, performance can vary considerably, depending on how many targets reside at common mail servers. Therefore, we examine its upper bound, where each target email resides at the same mail server. In this scenario, which we term *spam_upper*, mobile bots first connect to an open AP (STAGE 1). Because bots only connect to a single mail server, we assume its IP address is included in the setup stage, allowing bots to skip STAGE 2 and immediately begin establishing a TCP connection with the mail server (STAGE 3). Bots then proceed as normal through STAGES 4 and 5. At STAGE 6, they add as many recipients as possible, such that they can finish STAGES 7 through 9 before losing connectivity with the current open AP; the attack then continues from STAGE 1 with the next available open AP. In determining the number of recipients to add at each AP, this technique achieves a level of precision nearly impossible in practice, allowing us to confidently identify the optimal amount of spam our mobile botnet could send.

For both the spam_lower and spam_upper attacks, we use the average email address length of 23 bytes for our target emails and the dominate domain length of 11 bytes for our mail servers [3]. Spam message data is set to 5KB, which was the size of

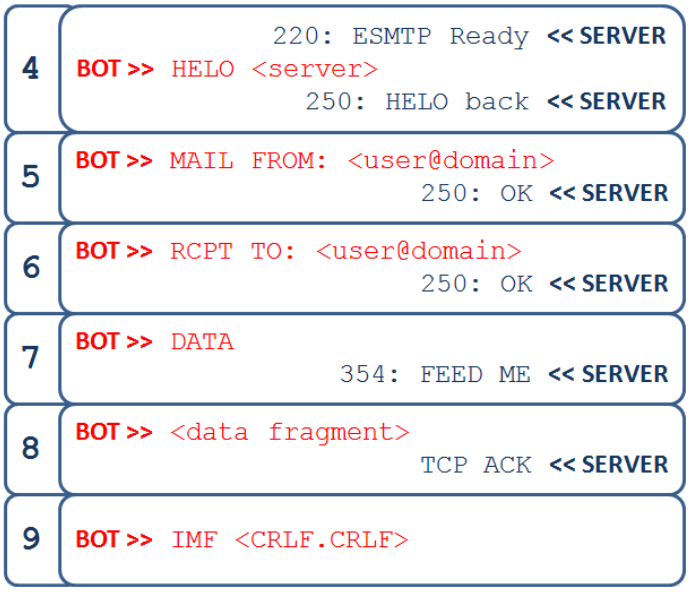


Figure 5.2: SMTP portion of mobile bot's SPAM protocol

over half of all spam emails for the first half of 2010 [8]. Since Ethernet's MTU is typically 1,500 bytes, bots can't send the entire 5KB message payload as a single data fragment in STAGE 8, requiring them to repeat the stage four times before completing the protocol at STAGE 9.

5.5 Experimental Results

In this section, we examine the results of our simulated mobile botnets' activities. Our simulator makes use of fine-grained cab and bus mobility traces, as described in Section 5.4.1. Ignoring the issue of infection (Section 5.3.3), we treat each cab/bus as a mobile bot. We examine the feasibility of mobile WiFi botnets during weekends and weekdays; for the cab traces during weekdays, we only examine the the office-commute hours (i.e., rush hours) of 6–10 a.m. and 3:30–7:30 p.m. For buses, we will also explore the effect of our various AP-selection algorithms on the different botnet attacks. For each AP-selection algorithm and attack, we are interested in determining both how successful the attack is and how distributed it is across open WiFi APs, further hindering detection.

5.5.1 AP-Selection Algorithms

First, we will examine the performance of the various AP-selection algorithms defined in Section 5.4.3 independent of the attack protocols. We examine each algorithm in terms of its cumulative throughput, demonstrating its potential achievable"work", and its APs' connection durations, demonstrating how quickly it switches between APs, which increases detection difficulty. Figures 5.3 and 5.4 show the cumulative throughput for each AP-selection algorithm for weekdays and Sunday; we have omitted Saturday since its results are similar to Sunday. From the figures, we find that the cumulative throughput displays a clear diurnal trend, with peak throughput achieved at around 5 p.m. on both weekdays and Sunday (i.e., weekends). This increases in throughput is a consequence of the number of buses in service at that time of day, with the fewest buses in service during the early morning and late-evening hours. The number of buses in service, and thus total throughput, increases considerably during the morning rush hours of 5–8 a.m. for weekdays. During weekends, when as many buses aren't in service during the early morning, we observe a smoother increase in throughput. For both weekends and weekdays, the total number of buses and throughput increases until the evening rush hour of 5 p.m. before decreasing once again. Furthermore, as the day progresses, the difference in throughput for the various AP-selection algorithms becomes more pronounced, culminating at around 5 p.m.. At 5 p.m. on weekdays, each improved selection algorithm increases the throughput by an average $\approx$57 million Mbit/s, while on weekends, the average improvement is $\approx$46 million Mbit/s.

These deviations result from the different approaches taken in minimizing AP switches and distance. Figures 5.5 and 5.6 show the CDF of the APs' connection durations seen over an entire weekday and Sunday for the different AP-selection algorithms, respectively. We find that the the intelligent AP-selection algorithms produce nearly identical results, which are clearly different than those of the Naïve algorithm.

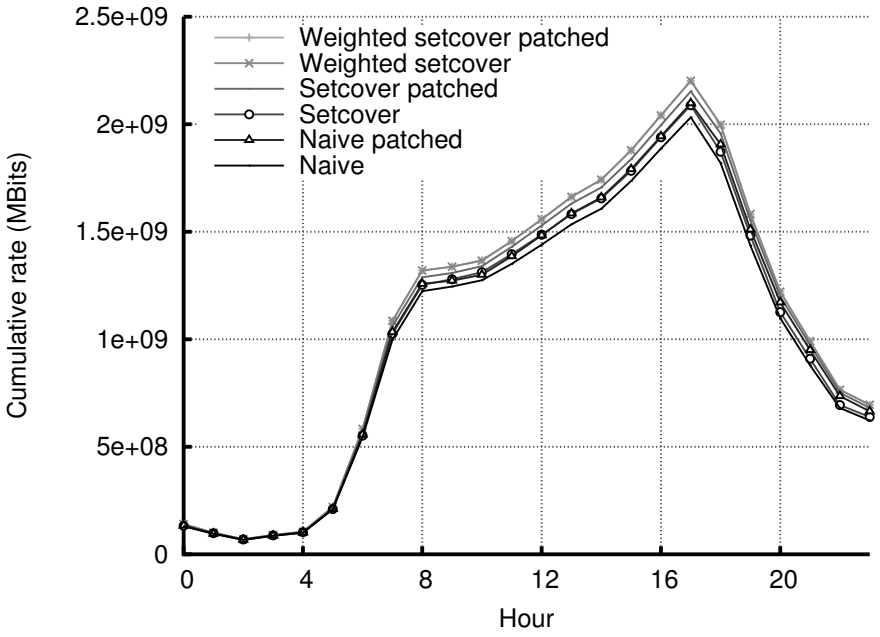


Figure 5.3: Cumulative throughput during weekdays

This is due to the intelligent algorithms' attempts to minimize the number of AP switches, which results in longer connection durations with most APs. For example, ≈4% fewer bots have a connection duration over 30 seconds for the Naïve algorithm than the intelligent algorithms. Interestingly, we find that the two curves intersect at about 8 seconds, indicating that the intelligent algorithms have a greater number of short-lived connections (i.e., less than 8 seconds) than the Naïve approaches. After having selected those APs with the longest durations to minimize AP switches, the intelligent algorithms select whichever short-lived connections remain, such that all possible seconds are covered by some AP within range. As a result, they have a greater amount of both short-lived and long-lived connections, producing the observed trend. Despite these minute variations, all AP-selection algorithms are fairly distributed. For example, 25% of all AP connections are for 8 seconds or less, while ≈98% are less than a minute. This means the majority of attacks utilize each AP for less than a minute, significantly complicating detection.

Since the connection durations are nearly the same for the intelligent algorithms,

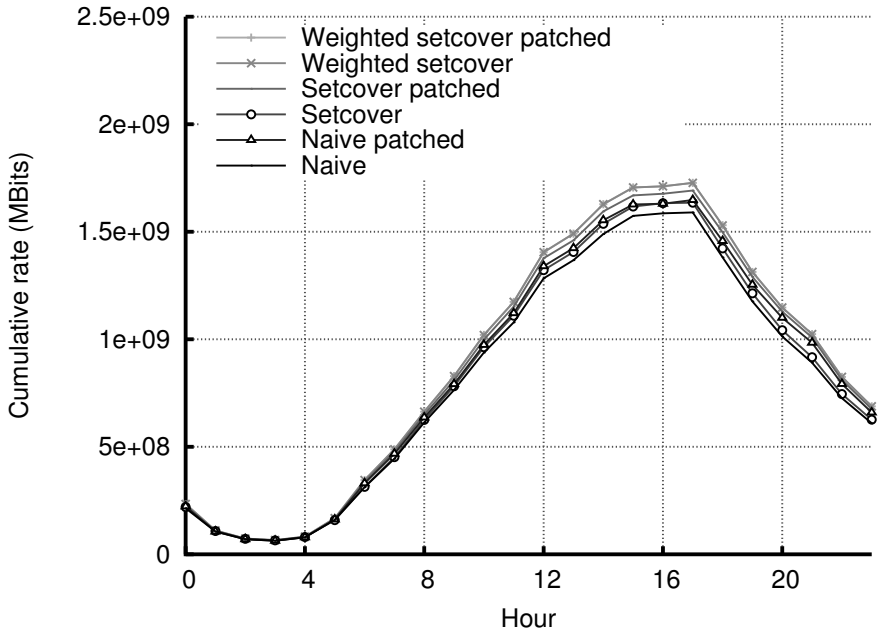


Figure 5.4: Cumulative throughput on Sunday

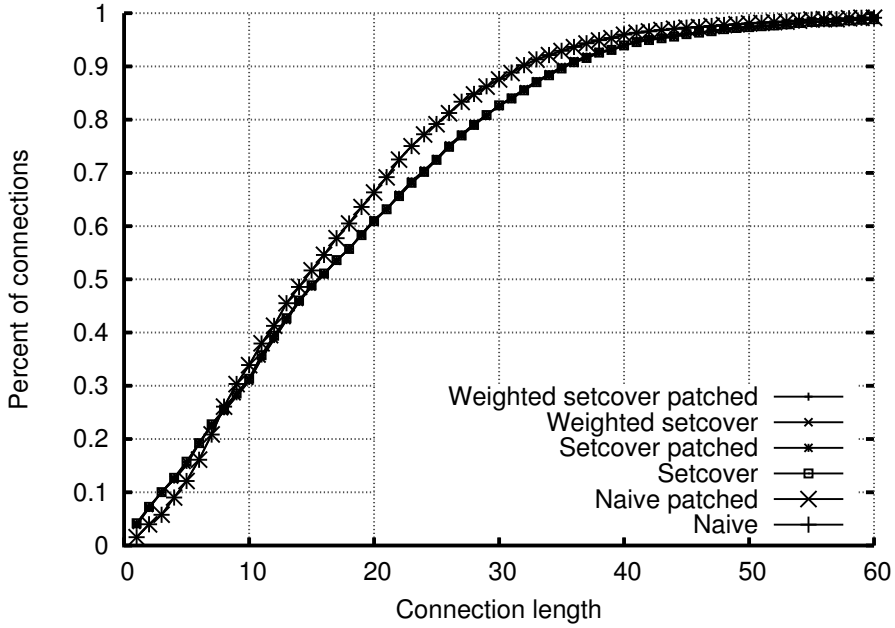


Figure 5.5: CDF of APs' connection durations per day during weekdays

they cannot account for the variations in throughput like they can for the Naïve approaches. Rather, these deviations are a consequence of how the intelligent algorithms attempt to minimize distance. Because Setcover doesn't take the AP's distance into

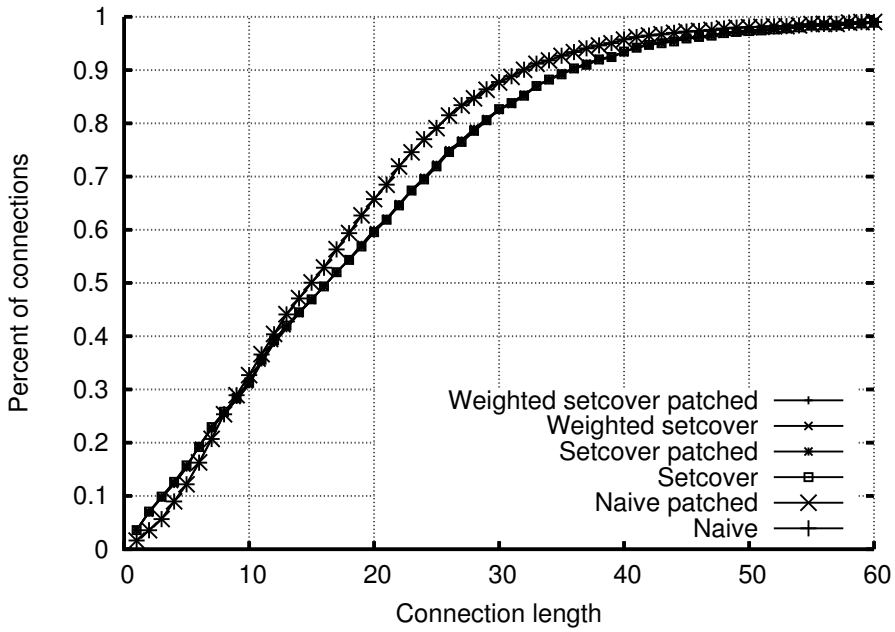


Figure 5.6: CDF of APs' connection durations per day on Sunday

account, it performs the poorest. Naturally, the patched version of Setcover—which optimizes distance (and thus throughput) at bus stops—performs slightly better, and the Weighted Setcover algorithms—which take distance into consideration during every AP selection decision—achieve the best throughput.

Lastly, Fig. 5.7 shows a CDF of the APs' connection durations by hour for Weighted Setcover on weekdays.[8] Each hour is represented in the plot, with some interesting hours labeled for analysis. From the figure, we can see that the APs' connection durations also follows a diurnal trend. In the early morning and late-evening hours, when there are fewer buses and vehicles on the road, bots maintain shorter connections with each AP; for example, at 3 a.m. and 11 p.m., ≈98% and ≈92% of AP connections last 30 seconds or less, respectively. During rush hours and midday, the connection durations increase, with 8 a.m., 12 p.m. and 5 p.m. having ≈85%, ≈80% and ≈78% of connections lasting 30 seconds or less, respectively. Despite these incongruities, ≈98% of all connections are for a minute or less, meaning attacks will

[8]The various intelligent algorithms all have similar results, and even Naïve demonstrates similar trends, though for slightly decreased connection durations.

be spread across many different APs, hindering detection.

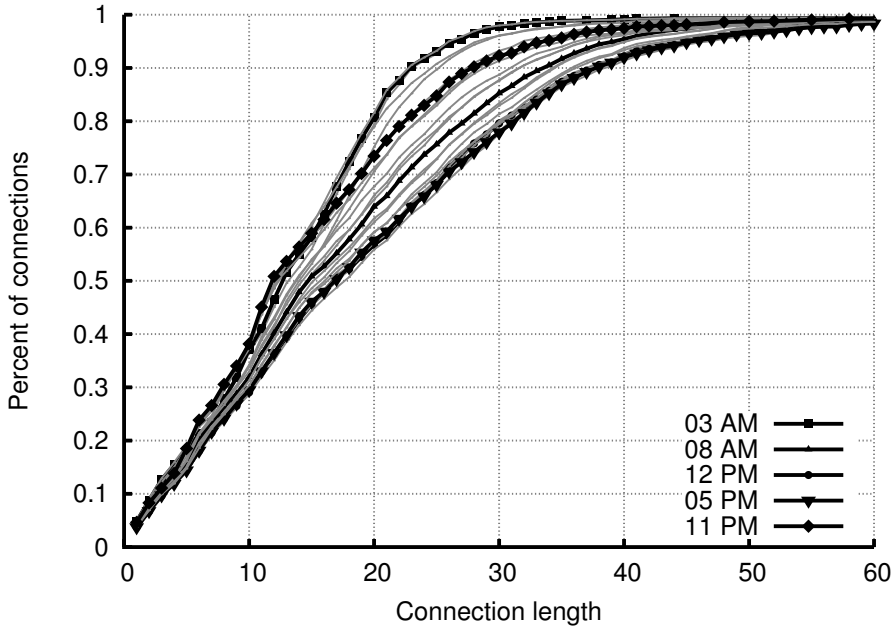


Figure 5.7: CDF of APs' connection durations per hour for Weighted Setcover during weekdays

5.5.2 Command and Control

In this section, we aim to answer the following questions concerning mobile WiFi botnet C&C:

- What level of control does the botmaster have in terms of the number of bots reachable and how frequently the bots can receive commands?
- How long does it take a command to propagate through the reachable botnet?
- How is the botnet distributed across open WiFi APs?

First, to help determine the botmaster's level of control, we calculated, for each hour, the average number of unique bots (i.e., cabs or buses) that could receive *at least one* command, which we term the *reachability*. For cabs during the weekday rush hours, the results were fairly consistent, with typically ≈73–75% of the total

cabs receiving commands. Exceptions included 6–8 a.m., with only ≈56–65%, and 6:30–7:30 p.m., where the amount steadily increases from ≈75% to ≈82% as the week progresses, presumably from people also going out recreationally in the evening (e.g., dinner). During the weekend, the results demonstrate a clear diurnal trend, shown in Fig. 5.8. Unsurprisingly, Saturday evening from 7 p.m. to midnight—when people go out for dinner or to bars—provides botmasters access to the largest portion of their available botnet per hour (≈75–84%). These results are also observed for Saturday and Sunday mornings between midnight and 2 a.m., when people are returning home from a Friday and Saturday night out.

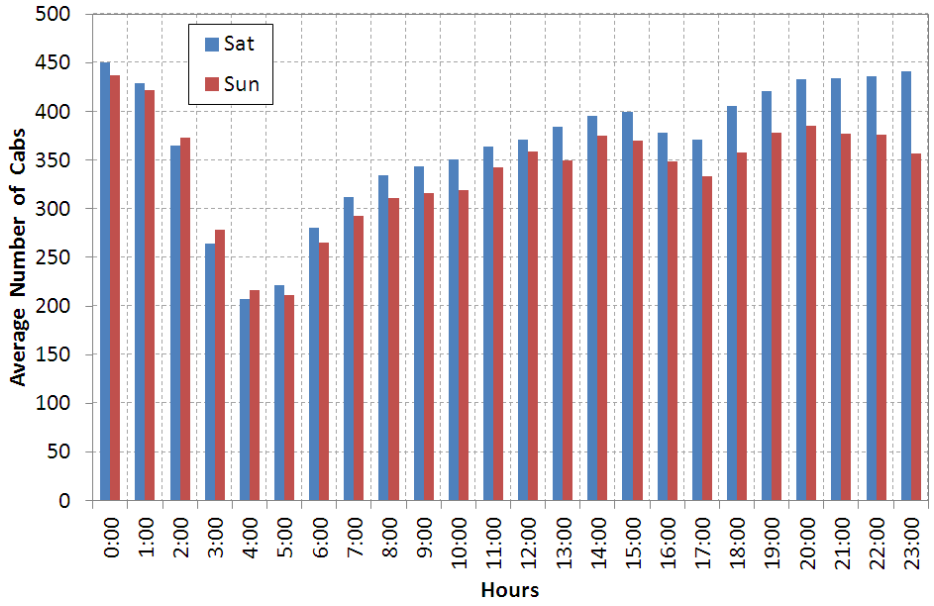


Figure 5.8: Average number of cabs receiving commands on weekends

When analyzing the bus botnet's C&C results, it quickly became apparent that the various AP-selection algorithms achieved nearly identical results. To determine if this was a shortcoming of the selection algorithms or an artifact of the bots' mobility and San Francisco's open WiFi topology, we simulated a theoretically *optimal* AP-selection algorithm that is impossible to achieve in practice: we treat every AP within range of the bot as the *same* AP at a distance of 0 meter from the bot. Notice

that in this scenario, bots will not need to perform AP switches unless there is a period when no APs are within range; the throughput at every AP will also be the maximum achievable. By removing any overhead due to AP reassociations and using a maximum throughput at all times, we can determine if there is any room for potential improvement in our AP-selection algorithms with respect to C&C.

As previously done for cabs, we calculated and plotted the bus botnet's reachability, shown in Figs. 5.9 and 5.11 for weekdays and Sundays; Saturdays have been omitted due to their similarity to Sundays. For both weekdays and weekends, the plots demonstrate the same strong diurnal trend—dependent on the number of buses in service—as in Figs. 5.3 and 5.4. Since the number of buses in service changes hourly, we have plotted the percentage of running buses reachable per hour in Figs. 5.10 and 5.12. From the figures, we can see that the various AP-selection algorithms achieve virtually identical results, which are nearly those achieved by the optimal algorithm. These results indicate that achieving optimal hourly C&C is independent of the AP-selection algorithm used. This is because botnet C&C can function effectively when receiving commands every few hours, and for a given hour, it is highly likely that bots will be within range of at least one AP long enough to receive a command. Form Figs. 5.10 and 5.12, we can see that for any hour of the day, more than 80% of the running buses can receive a command, often more than 90%.

Next, for those bots able to receive commands within a given hour, we calculated the percentage of minutes when a new command could be received, which we term the bot *responsiveness*. It can be interpreted as a bot's probability—for a given hour—of receiving commands within a minute of them being issued. For the less regular cab mobility, the botnet's average responsiveness is shown in Figs. 5.13 and 5.14 for weekday rush hours and weekends; a complementary CDF of the responsiveness per bot across all weekdays and weekends for particular hours is shown in Fig. 5.15. From the figures, we see that the evening hours afford the greatest responsiveness due to

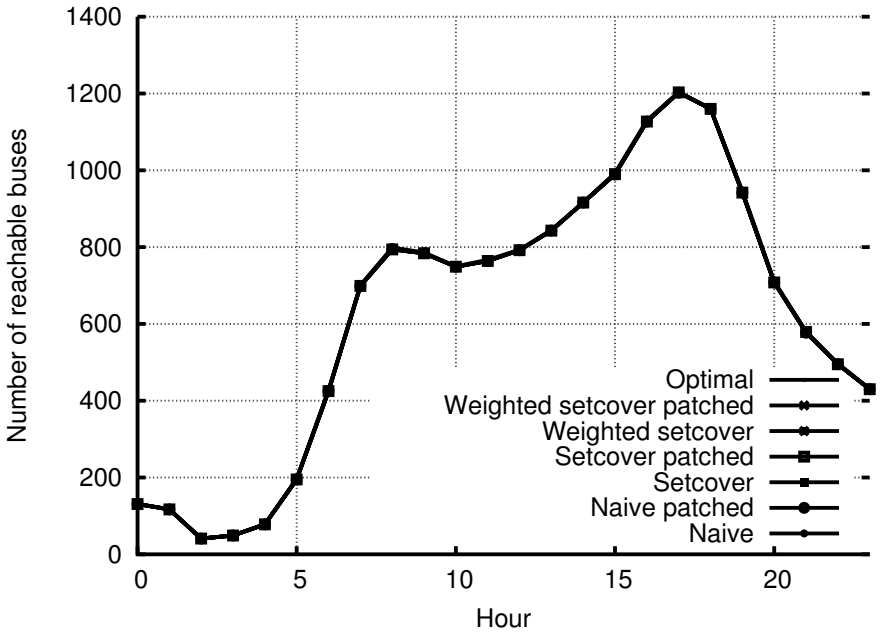


Figure 5.9: Number of reachable buses per hour during weekdays

the combined traffic congestion of people returning from work and going out for the evening. When traffic is less congested, cabs are able to move more quickly, spending less time at each WiFi AP. Once out of an AP's range, bots must associate and reconnect with a new network before receiving commands, and this delay results in the lower average responsiveness demonstrated during less popular commute hours. From Figs. 5.8 and 5.14, which share a strong diurnal trend, we find that the peak evening and early morning hours not only grant access to a large percentage (≈75–84%) of the available botnet, but that bots receiving commands during these times are highly responsive, with the average responsiveness exceeding 50%. This fine-level of control is also observed during weekday evenings between 6:30–7:30 p.m. While evenings impart the greatest level of control during a given hour, all the hours examined provide relatively quick control over a significant portion of the botnet. Even during the worst period, between 4–6 a.m. on weekends when people are asleep and fewer cabs are active, ≈39% of mobile bots are able to receive commands with an average responsiveness above 30%. If a greater level of control is necessary, botmasters

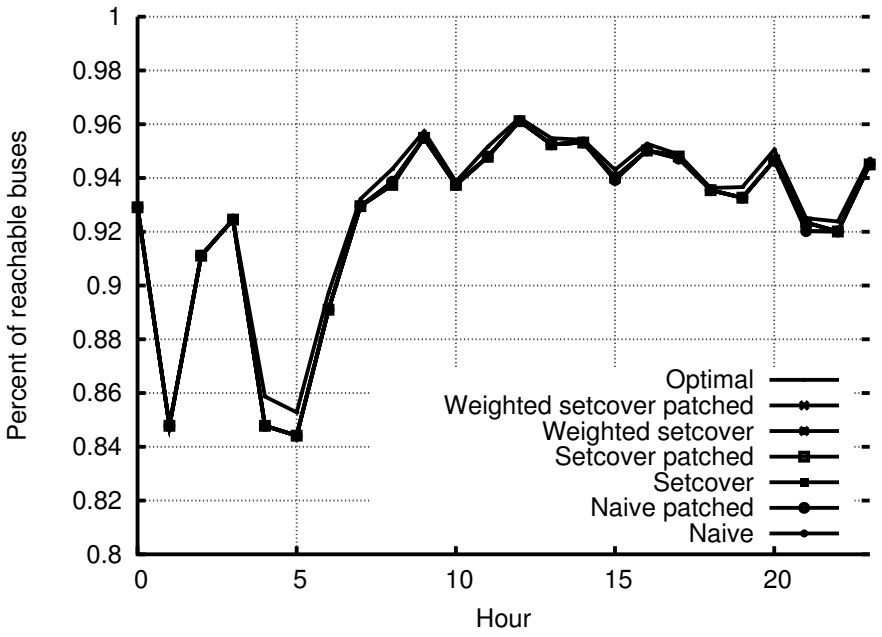


Figure 5.10: Percentage of active buses reachable per hour during weekdays

could resort to using home WiFi networks at this time, though it may increase their risk of detection.

Similar responsiveness results are observed for the bus botnet, shown for the various AP-selection algorithms during weekdays and Sundays in Figs. 5.16 and 5.16, respectively; again, Saturday is omitted due to its similarity to Sunday. From the figures, we find that, as with reachability, the various AP-selection algorithms have little effect on the bots' responsiveness, with all of them performing nearly optimally. Furthermore, with the exception of midnight, 1 a.m. and 5 a.m., the botnet's average responsiveness is at least 30% and often more than 35%. Figure 5.18 shows the CDF of the responsiveness per bot across all weekday hours when using Weighted Setcover; the other AP-selection algorithms produced similar results. From the figure, we find that for all the hours except midnight, 1 a.m. and 5 a.m., ≈20–40% of the botnet has a responsiveness greater than 50%. Taken in conjunction with our previous reachability results, we discover that, for any given hour, more than 80% of our bots will receive a command, and of those that do, typically more than 1/3 of them will receive

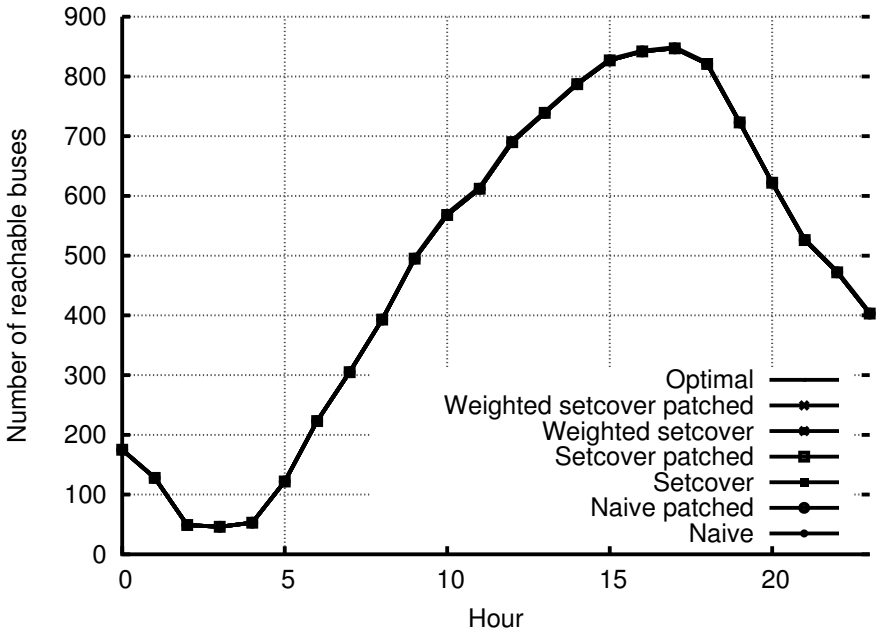


Figure 5.11: Number of reachable buses per hour on Sunday

the command within a minute of it being issued.

To determine how quickly commands spread throughout the botnet, we issue a new command at various hours, plotting how many bots the command propagates to over a 2-hour period. For the cab botnet, we show this for 4 command-injection times averaged across all weekdays and 4 command-injection times averaged across all weekend days in Fig. 5.19. Fig. 5.20 shows 4 weekend command-injection times independently averaged across all Saturdays and Sundays. In Fig. 5.19, all commands (except those issued at 7 a.m. on a weekend) reach at least 300 mobile bots (i.e., ≈56% of the botnet) within 30 minutes of injection; after 2 hours, they have propagated to at least 75% of the reachable botnet. Similar results are observed in the more detailed weekend plot. It is apparent that certain injection times provide significant gains in command propagation. Issuing commands at 10 p.m. on weekends or at 9 a.m. or 6:30 p.m. on weekdays results in at least 65% of the reachable cabs receiving the command within 30 minutes of injection. From Fig. 5.20, we find that commands issued on Saturdays at 10 p.m. propagate even more quickly, arriving at ≈76% of the

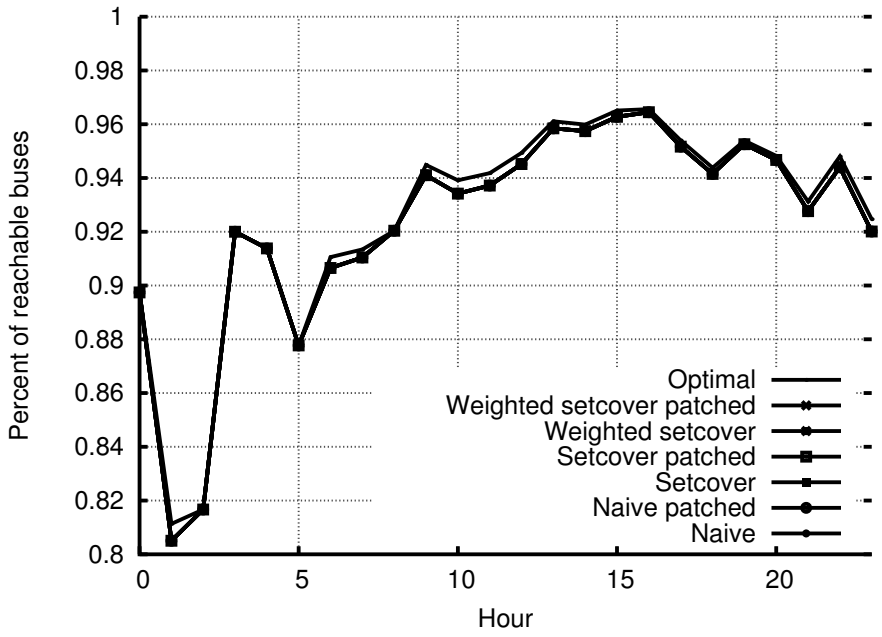


Figure 5.12: Percentage of active buses reachable per hour on Sunday

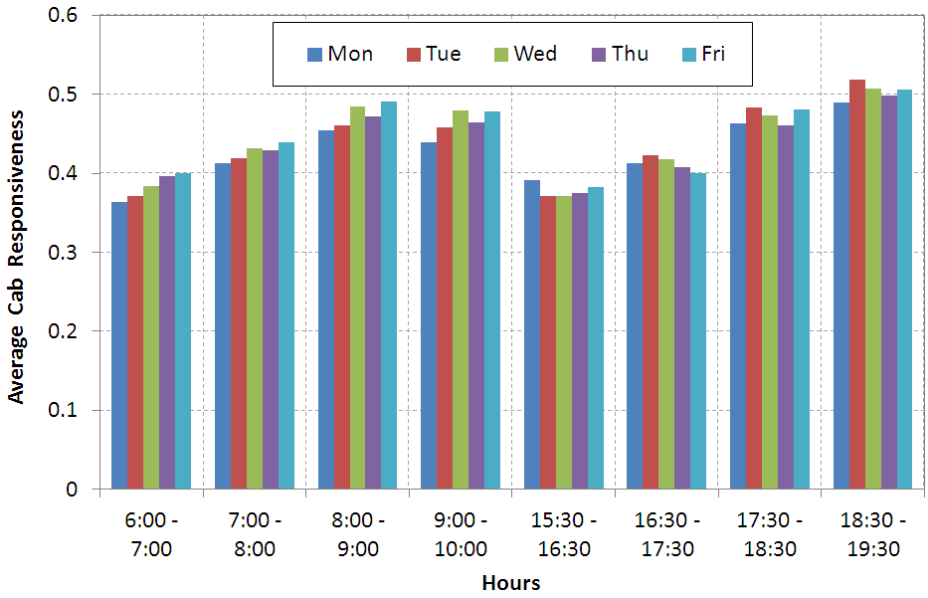


Figure 5.13: Average cab responsiveness during weekday rush hours

reachable botnet within 30 minutes of injection. Even in the worst case, when issuing a command at 7 a.m. on a weekend, over 61% of bots have received the command within 2 hours of injection.

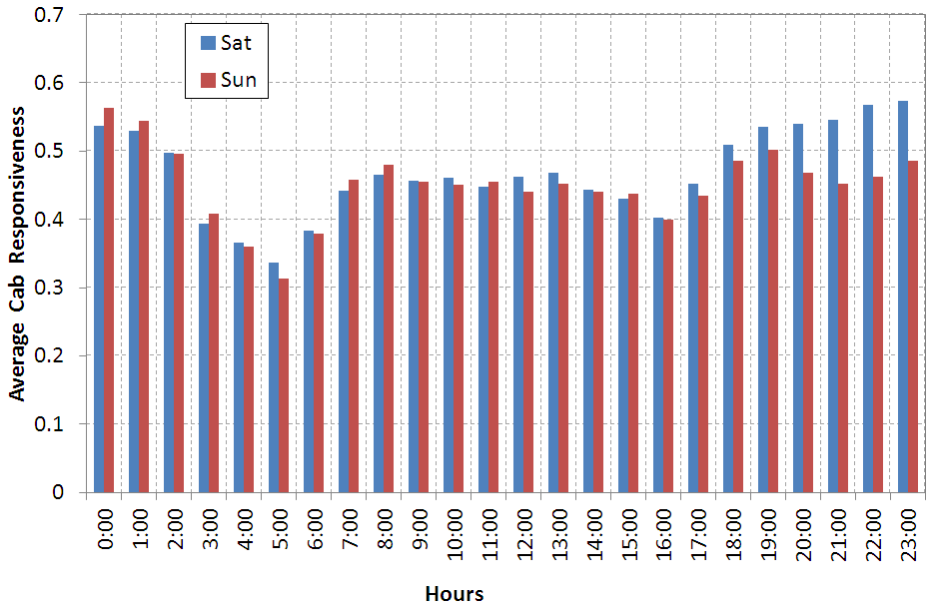


Figure 5.14: Average cab responsiveness during weekends

For the bus botnet, we look at the number of active buses and the number able to receive commands for the different AP-selection algorithms and injection times, shown in Figs. 5.21 and 5.22 for 4 and 8 a.m. on weekdays, respectively. We observed similar behavior for all injection hours on weekdays and weekends, with most exhibiting convex growth, as in Fig. 5.22, while the early morning hours demonstrated concave growth, as in Fig. 5.21. In both figures, the upper line represents the number of buses that have been active since the time of injection, while the lower line shows the number of bots that have received a command since injections for the various AP-selection algorithms. Notice in both figures that the shape of the curve is dictated by the upper line, i.e., the number of buses active and able to receive commands. The convex growth in early morning hours results from there being so few buses in service; when more buses are introduced, the relative growth rate is more extreme. From the figures, it is clear that the improved throughput of the more advanced AP-selection algorithms has almost no influence on command propagation. Despite the choice of AP-selection algorithm or the time of injection, commands quickly reach over 80% of the botnet—and soon over 90%. For example, at 8 a.m. on a weekday, it takes

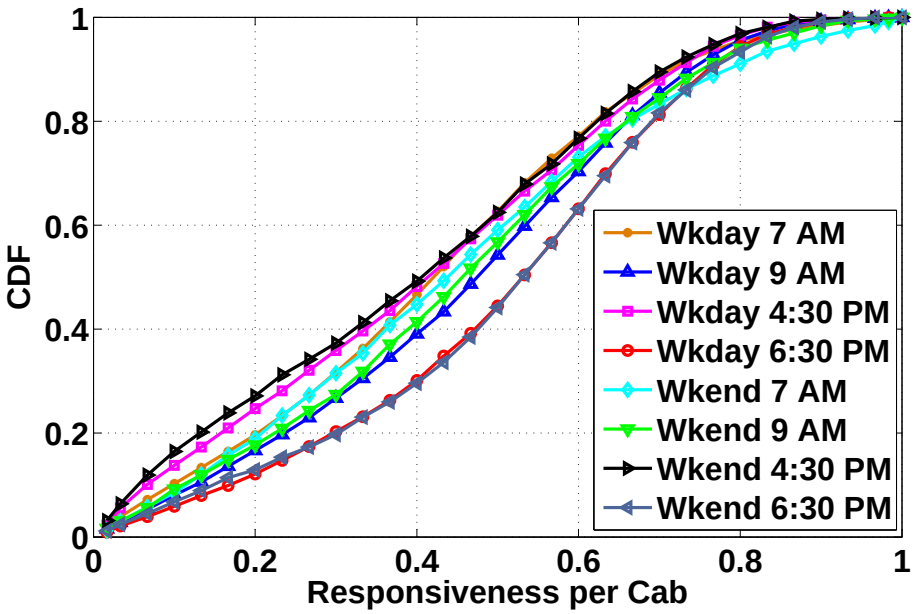


Figure 5.15: CDF of responsiveness per cab

only ≈10 minutes for the command to reach over 80% of the active buses. Despite the constant introduction of new buses, this ratio continues to improves with time, such that after one hour, ≈94% of active buses have received a command. These results demonstrate that botmasters can reach a large percentage of their botnet in a relatively short amount of time, and in the case of buses, this propagation keeps pace with the increasing size of the botnet as new buses are introduced.

Lastly, we are interested in determining how the botnet is distributed over the open WiFi networks. If only a small set of networks are used, then they have a significant view of the overall botnet activity, making detection and mitigation easier. Fig. 5.23 shows a heat map of the open APs used by the cab botnet from 6–10 a.m. on weekdays. Each dot is an area of 2,500 m^2, where the color represents the average number of unique cabs receiving commands from open APs in that area. Fig. 5.24 is a CDF plot that's complementary to the heat map, showing the average number of cabs using an open AP per minute. As can be seen from the map, the botnet C&C is spread across a large number of open WiFi networks, making detection difficult.

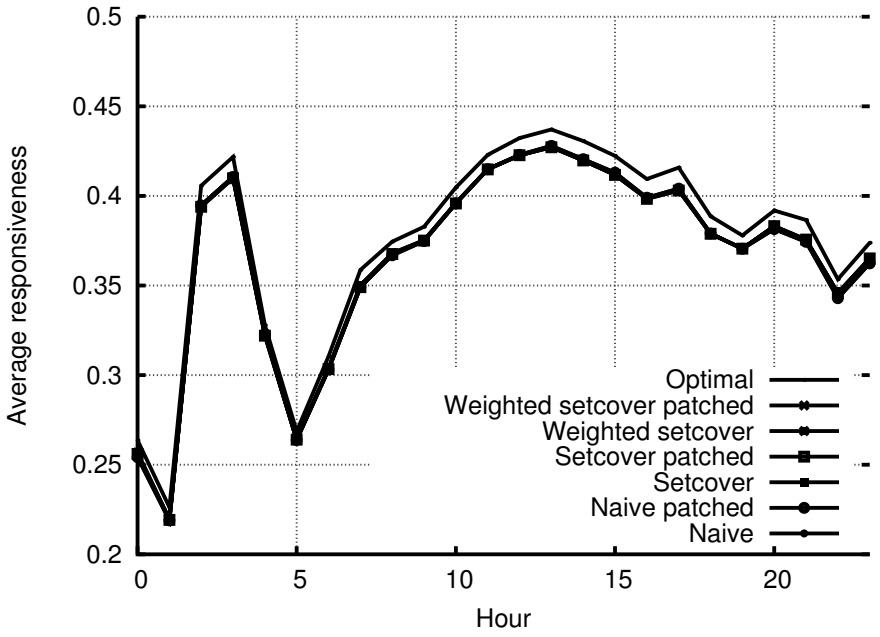


Figure 5.16: Average responsiveness per hour during weekdays for bus botnet

While the open APs along major highways and downtown are utilized by more cabs, most service only around 50 different cabs during the 4-hour period. Furthermore, Fig. 5.24 shows that 90% of the open APs are used by only 5 or fewer simultaneous bots during any given minute. Of the remaining 10%, less than 2% ever have more than 10 bots using them during any given minute, and even they never have more than 23. Since commands can take only seconds to receive, even 23 bots sharing an open WiFi network produce little traffic during a minute, easily blending amidst other traffic and hindering detection.

C&C activity is also well distributed for the bus botnet. Fig. 5.25 is a CDF showing the number of bots performing C&C at each unique AP per minute for Weighted Setcover on weekdays; results were similar for the weekends and the other AP-selection algorithms. From the figure, we find that during any given minute, ≈90% of APs have 5 or fewer simultaneous bots using them for C&C, and ≈99% have 10 or fewer. The presence of 5–10 bots performing C&C per AP is easily hidden amidst the network's normal traffic. Fig. 5.26 shows the number of unique APs used

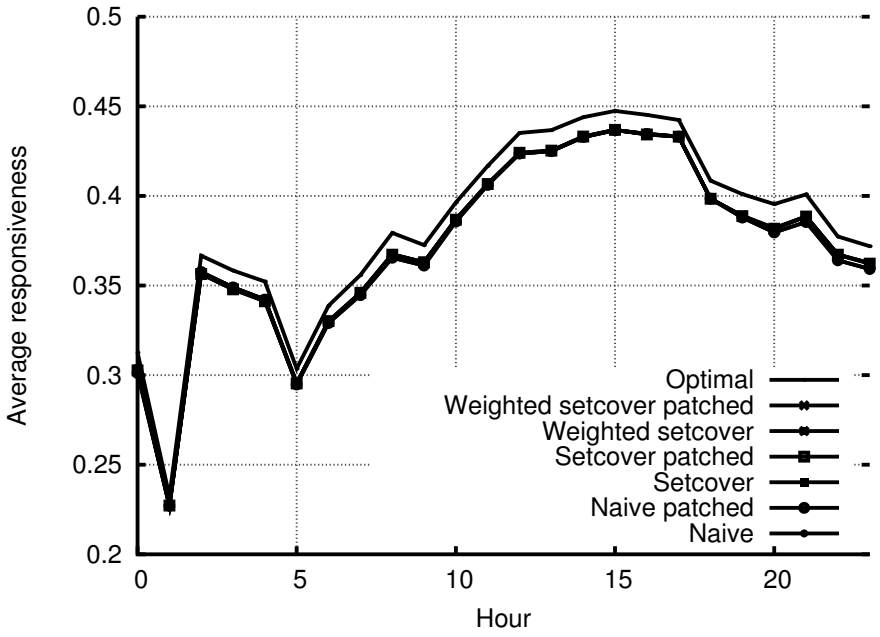


Figure 5.17: Average responsiveness per hour on Sunday for bus botnet

by bots per minute and hour for both the Weighted Setcover and the Naïve AP-selection algorithms. We find that there are typically 100–300 unique APs used per minute for C&C, sufficiently spreading the attack across multiple networks. While this number can occasionally drop to as low as 10 APs per minute in the early morning hours, we find, from Fig. 5.9, that the number of buses reachable during these times is also significantly reduced (to ≈50–100 bots), meaning that the attack is sufficiently distributed. We also find that Weighted Setcover uses fewer unique APs per hour than the Naïve protocol. This is an artifact of it minimizing AP switches by increasing the amount of time spent at each AP, resulting in the attack being spread across fewer open networks. Still, even intelligent AP-selection achieves a sufficiently distributed attack to make detection more difficult. From these results, we observe that a highly mobile WiFi botnet can successfully be controlled using only open WiFi APs, and when spread across many different APs, its C&C traffic for any given AP is small and easily hidden, making it difficult to detect.

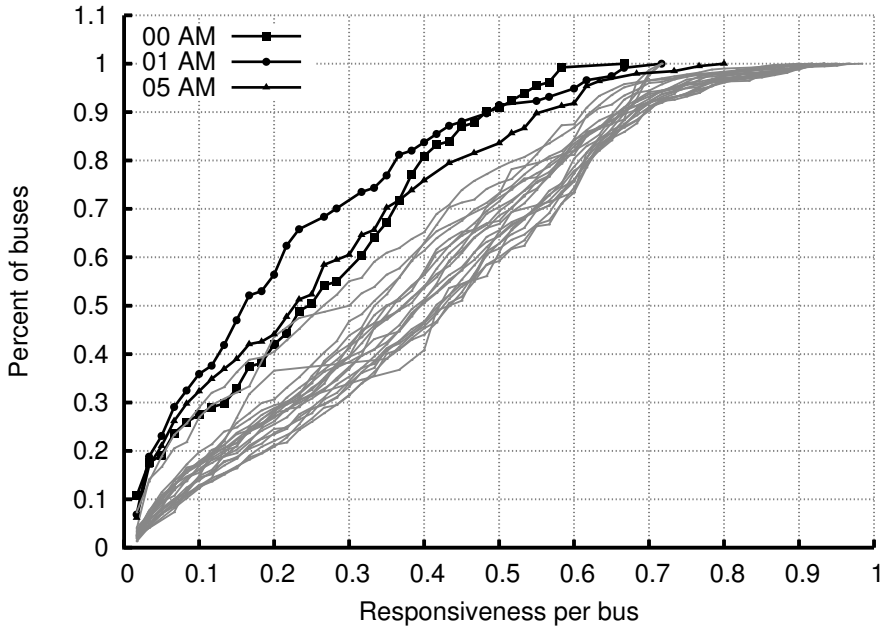


Figure 5.18: CDF of responsiveness for Weighted Setcover per hour during weekday for bus botnet

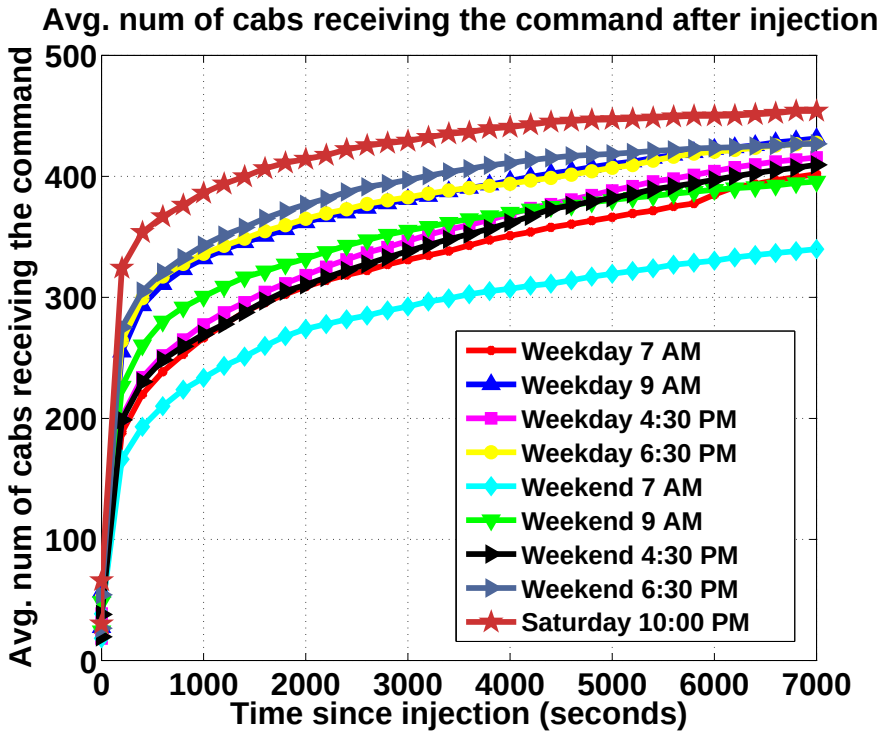


Figure 5.19: Command propagation for different injection times on weekdays and weekends for cab botnet

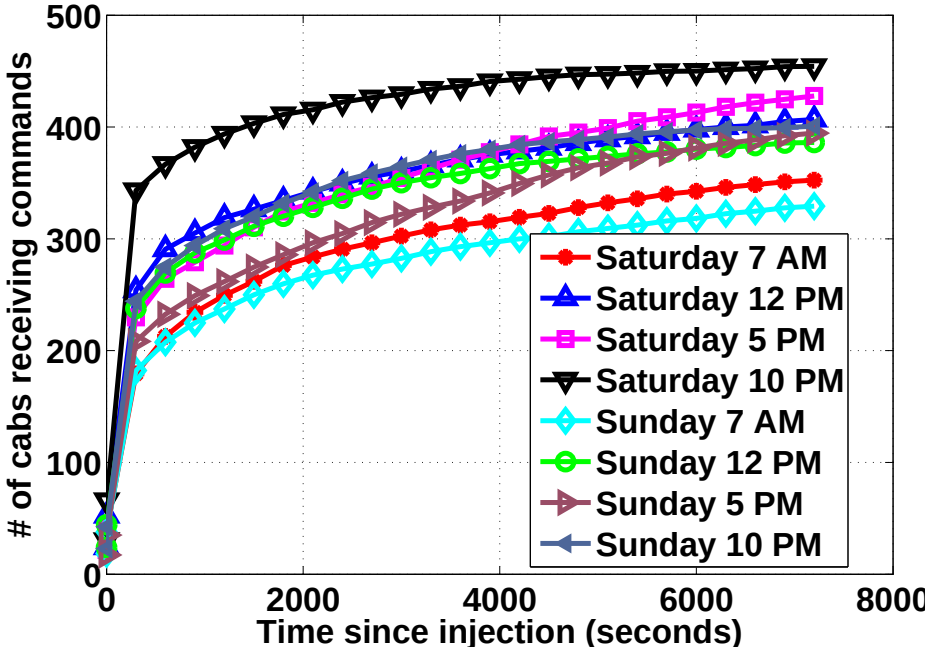


Figure 5.20: Command propagation for different injection times on Saturday and Sunday for cab botnet

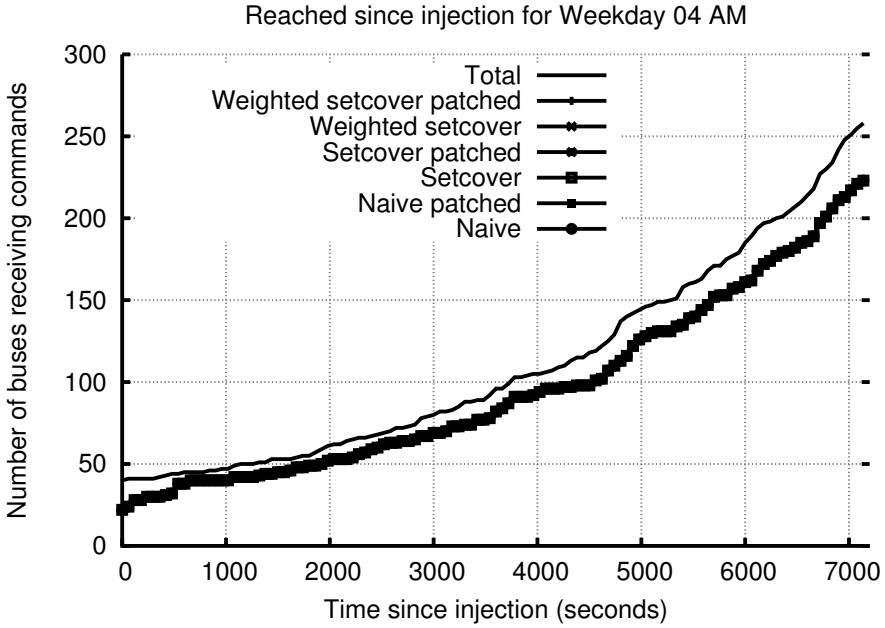


Figure 5.21: Maximum and actual command propagation for various AP-selection algorithms injected at 4 a.m. on weekdays for bus botnet

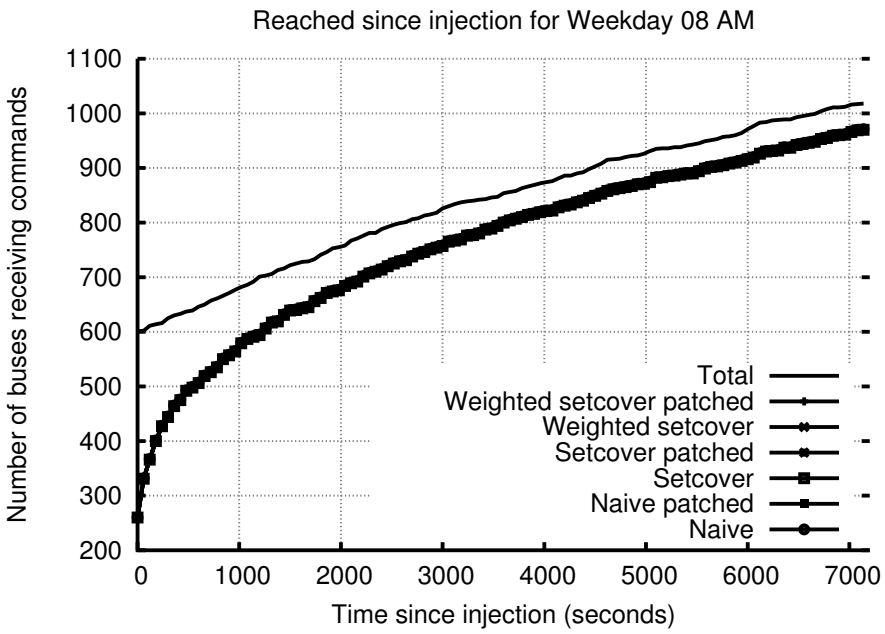


Figure 5.22: Maximum and actual command propagation for various AP-selection algorithms injected at 8 a.m. on weekdays for bus botnet

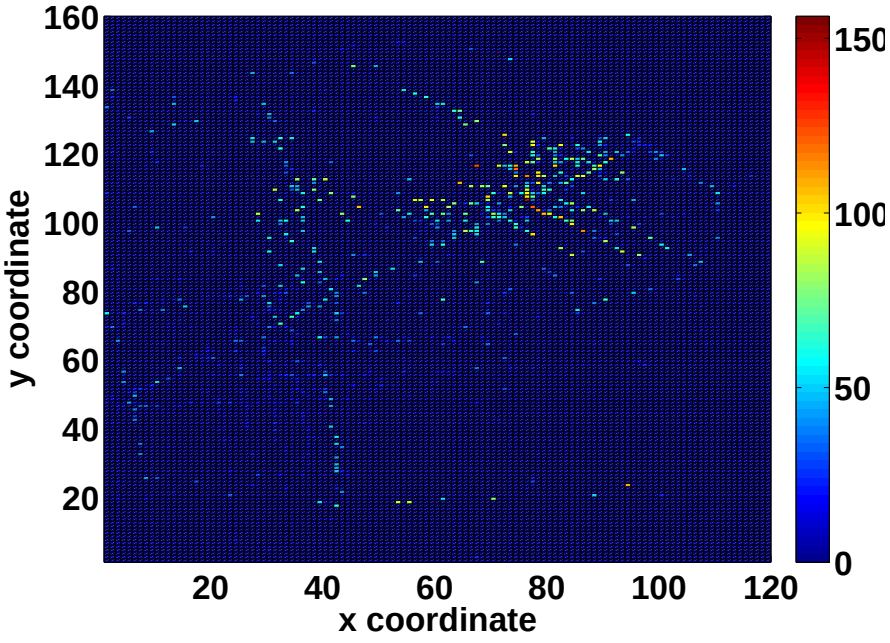


Figure 5.23: Heat map of open APs used during weekday morning rush hours by cab botnet

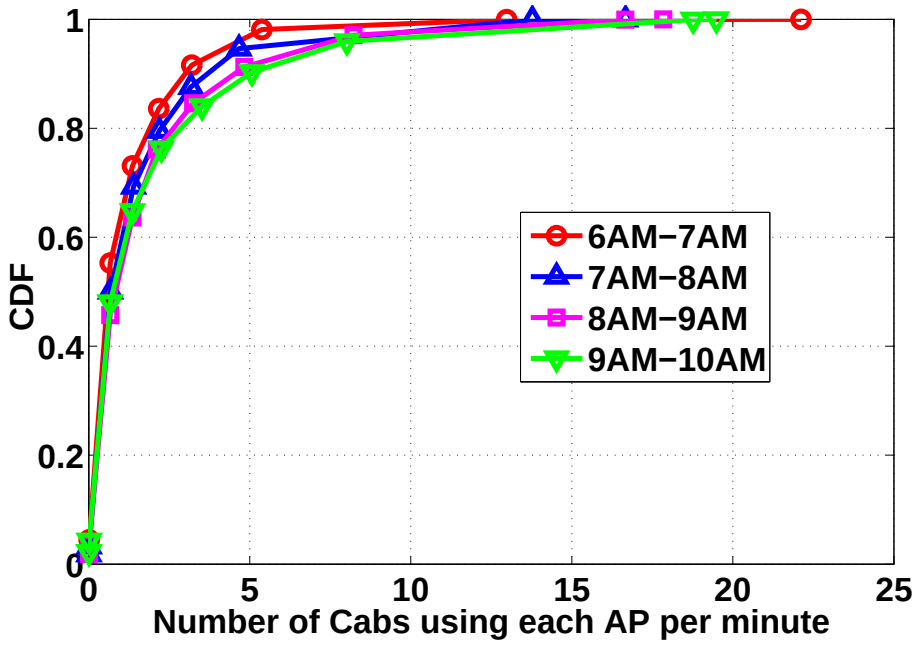


Figure 5.24: CDF of average number of cabs using open APs per minute for C&C during weekdays

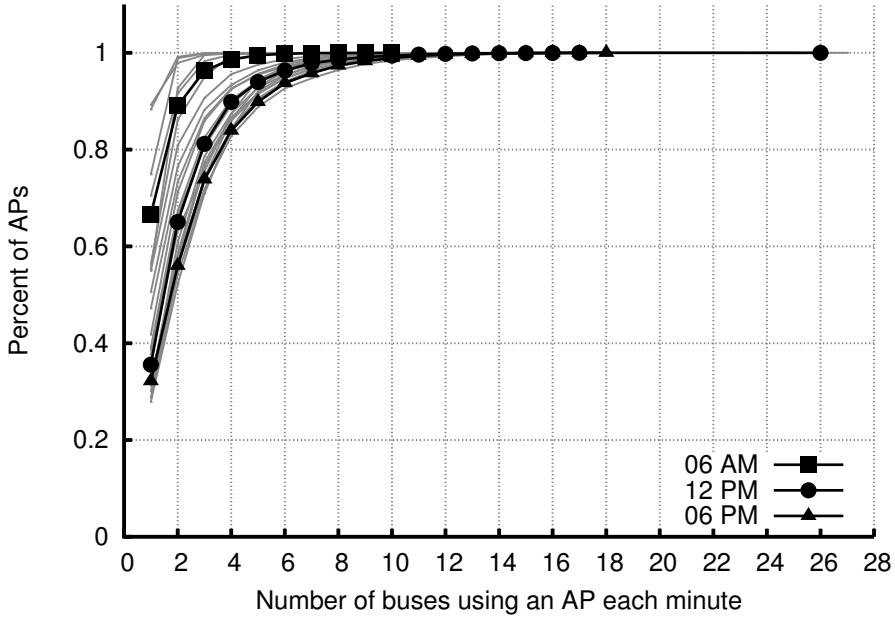


Figure 5.25: CDF of number of bots performing C&C at each AP per minute for Weighted Setcover during weekdays for bus botnet

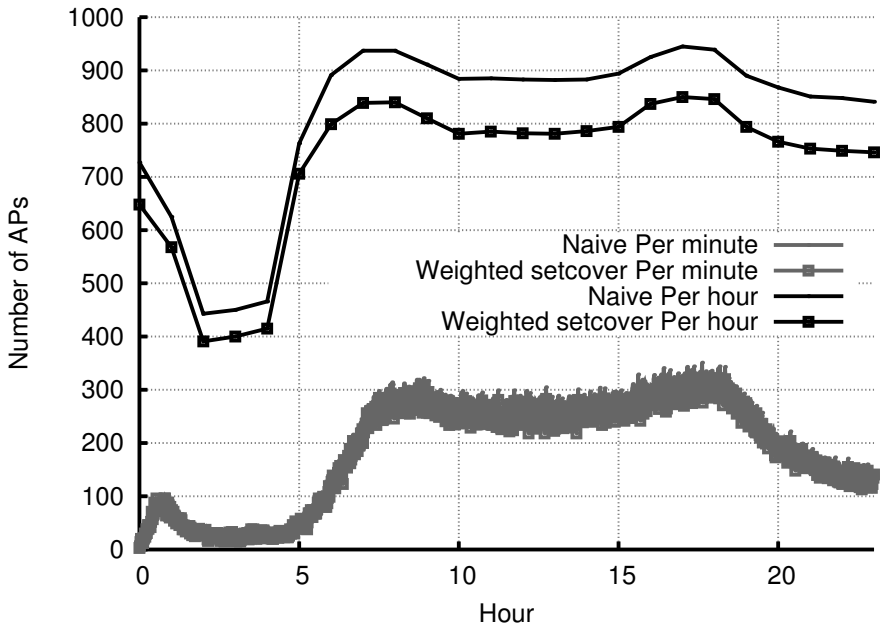


Figure 5.26: Total number of unique APs used in C&C per unit time during weekdays by bus botnet

5.5.3 DDoS Attack

In this section, we aim to answer the following questions concerning mobile WiFi botnet DDoS attacks:

- What is the botnet's capacity for DDoS attacks?
- How are DDoS attacks distributed across open WiFi APs?

To answer these questions, we simulate the DDoS protocol described in Section 5.4.4.2 using our cab and bus mobility data. Figures 5.27–5.30 plot the average number of SYN packets sent per hour by the mobile WiFi botnet for cabs during weekday rush hours and weekends and for buses during weekdays and Sundays.[9] In the case of the more random cab mobility, shown in Figs. 5.27 and 5.28, we find that during the peak weekday commute hours of 8–10 a.m. and 5:30–7:30 p.m., the botnet is capable of issuing ≈1.4 million SYN packets per hour (≈389 per second). During prime weekend hours, the capacity is even greater, achieving ≈1.4–1.7 million SYNs per hour (≈389–472 per second). Except for the lull between 3–7 a.m. on weekends, the mobile botnet can achieve between 0.8–1.2 million SYNs per hour (≈222–389 per second). Even at its lowest capacity, between 4–6 a.m. on weekends, the botnet can issue ≈500,000 SYNs per hour (≈139 per second).

For the more predictable bus mobility, shown in Figs. 5.29 and 5.30, we discover that the more intelligent AP-selection algorithms achieve a slight edge over their naïve counterparts. During the peak hour of 5 p.m., both Naïve and Setcover can issue ≈3.8 and 4 million SYN packets per hour (≈1,100 per sec) on weekdays, respectively, and ≈3 and 3.1 million per hour (≈850 per sec) on Sunday, respectively. Typically, all of the AP-selection algorithms can issue more than 2 million SYNs per hour (≈555 per sec) on weekdays and more than 1.5 million (≈417 per sec) on Sunday. Even at its lowest point, the botnet can issue ≈110,000 SYNs per hour (≈30 per sec) on

[9]Bus results for Saturday have been omitted since they are similar to Sunday's.

weekdays and Sunday.

According to [54], an unprotected server—or one using a default firewall configuration—can survive DDoS attacks of only 100 SYNs per sec. However, a properly configured firewall can survive DDoS attacks of 500 SYNs per sec, effectively defeating the DDoS capacity of the smaller cab botnet as well as the larger bus botnet during early morning hours. Nevertheless, we have demonstrated that mobile WiFi botnets have the potential to issue disruptive DDoS attacks; even with only a few bots or during sub-optimal hours, they could prove valuable when used to augment the DDoS attacks of larger, traditional botnets or mobile botnets in other cities, making them an ideal mechanism for infiltrating the mobile environment.

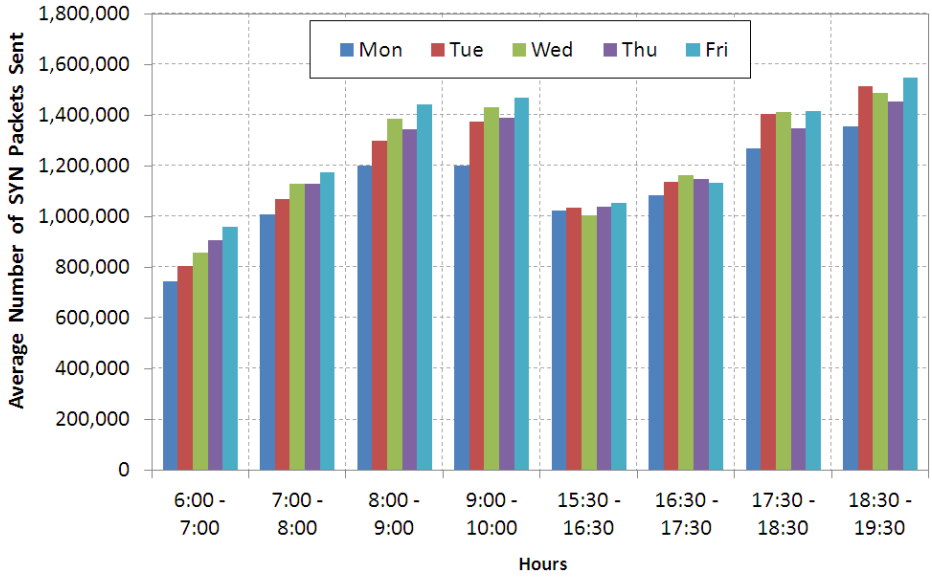


Figure 5.27: Average number of SYN packets sent during weekday rush hours by cab botnet

While mobile WiFi botnets' DDoS attacks possess sufficient capacity for mayhem, they must also be adequately distributed across multiple open WiFi networks to avoid detection. If the attack originates from only a small set of open APs, or too many bots operate from behind a single network, the attack is more easily detected and mitigated. To determine if the DDoS attack is sufficiently distributed to encumber detection, we examine the number of APs used per minute for both sets of mobility

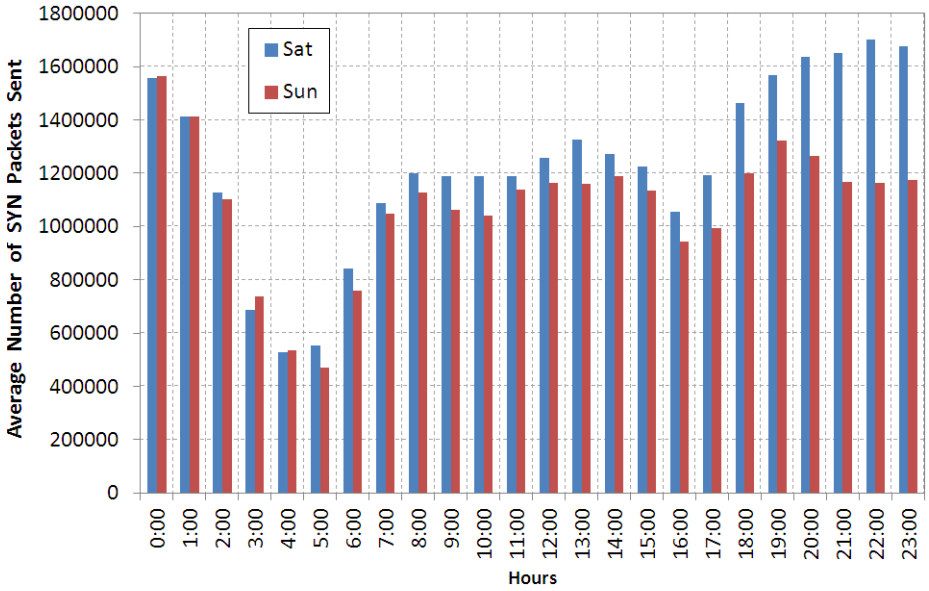


Figure 5.28: Average number of SYN packets sent during weekends by cab botnet

data. First, Fig. 5.31 plots the average and maximum number of APs used per minute by cabs during their morning commute hours (6–10 a.m.); a complementary CDF of the average number of cabs using an open AP per minute during this same time period is provided in Fig. 5.32. From Fig. 5.31, it is apparent that the DDoS attack is spread across multiple APs, ranging from 220 to over 340 different APs per minute in the average case. Furthermore, Fig. 5.32 shows that over 50% of the open APs used in the attack only service a single mobile bot per minute, and over 86% service 5 or fewer bots per minute.

To capture the extremes of the bus botnet's different AP-selection algorithms, we have plotted the number of unique APs used per minute and hour for Naïve and Weighted Setcover during weekdays in Fig. 5.33. As with C&C, the number of APs used in the DDoS attack demonstrates a strong diurnal trend in conjunction with the number of buses in service. From the figure, it is apparent that the DDoS attack is spread across numerous APs per minute and hour, with Weighted Setcover using fewer unique APs as a consequence of minimizing AP switches. From the CDF plot in Fig. 5.34, showing the the number of bots performing the DDoS attack at each AP

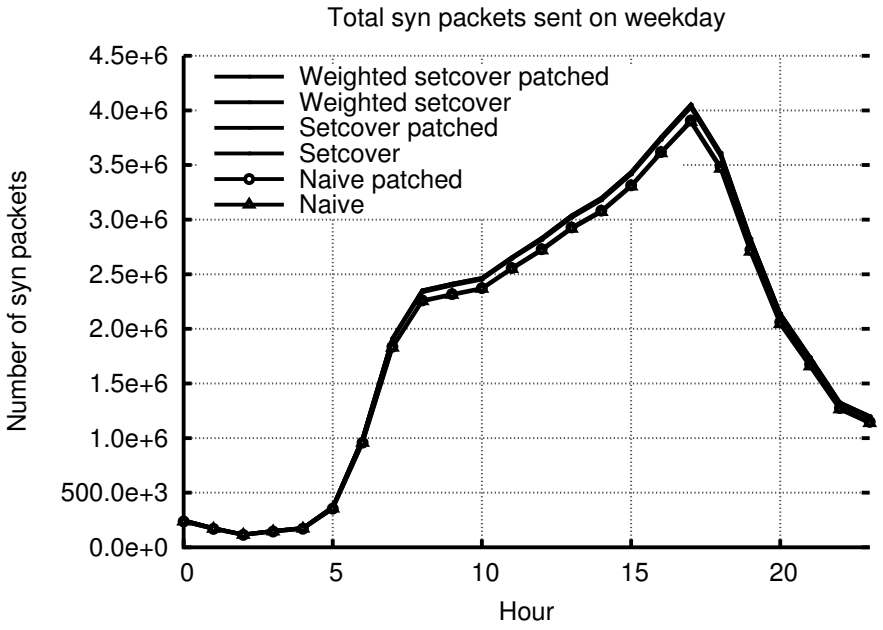


Figure 5.29: Number of SYN packets sent each hour during weekdays by bus botnet

per minute for Weighted Setcover during weekdays, we find, for all hours, that ≈90% of APs have 6 or fewer simultaneous bots per minute, and ≈99% of have 10 or fewer; these results are representative of the other AP-selection algorithms as well.

Thus, we find that mobile WiFi botnet DDoS attacks are distributed across many different open networks, with each network participating in only a small portion of the attacks. Obviously, this makes detection increasingly difficult for defenders. Considering their overall stealth and capacity, mobile WiFi botnets could prove a serious future threat as a DDoS attack mechanism.

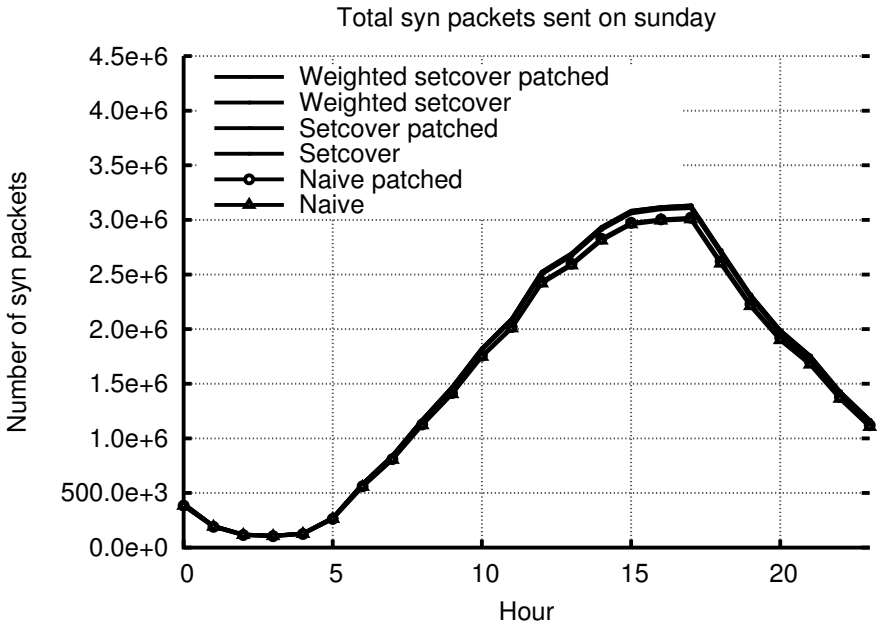


Figure 5.30: Number of SYN packets sent each hour on Sunday by bus botnet

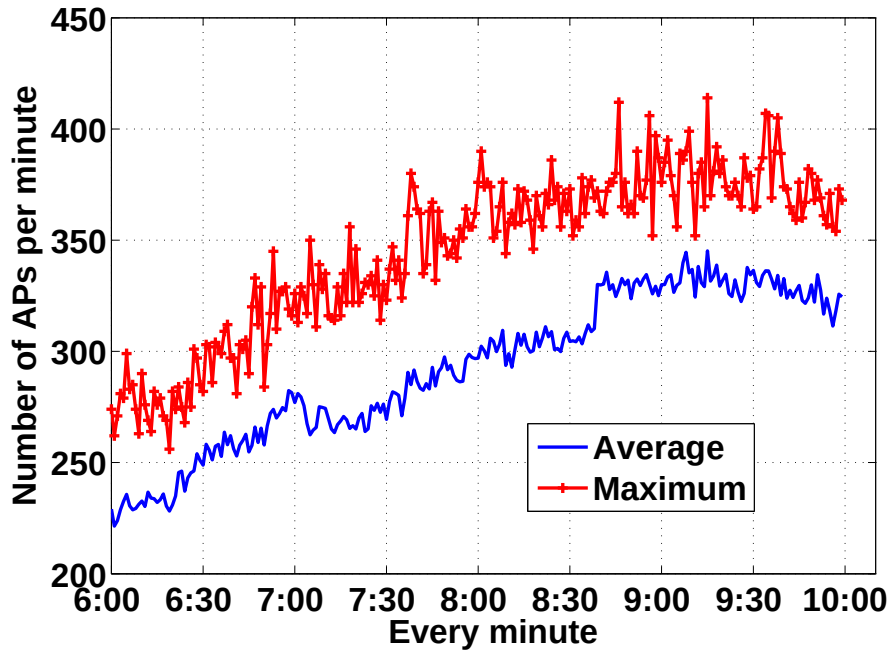


Figure 5.31: Average and max number of APs used per minute for DDoS attacks during weekday morning rush hours by cab botnet

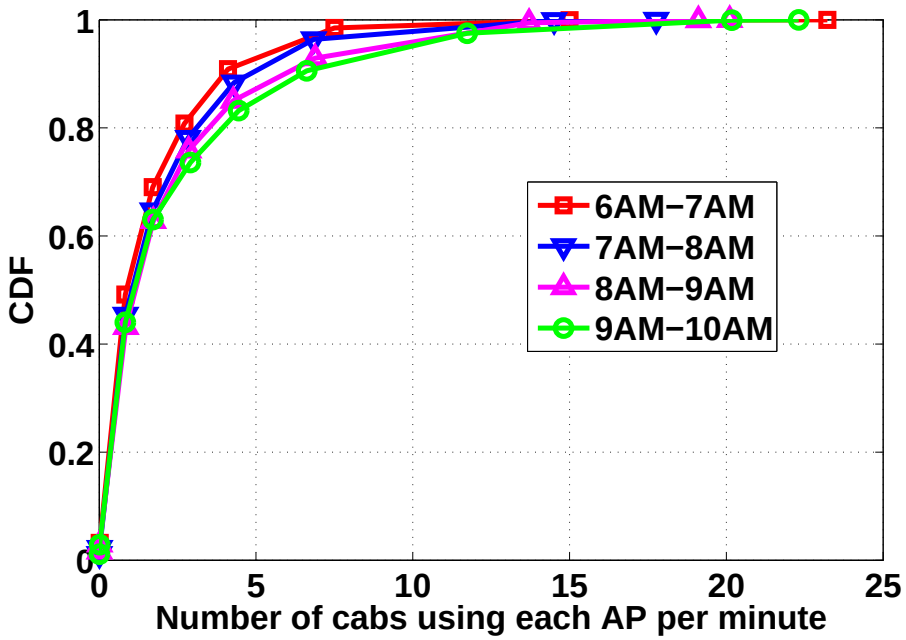


Figure 5.32: CDF of average number of bots using open APs per minute for DDoS during weekdays for cab botnet

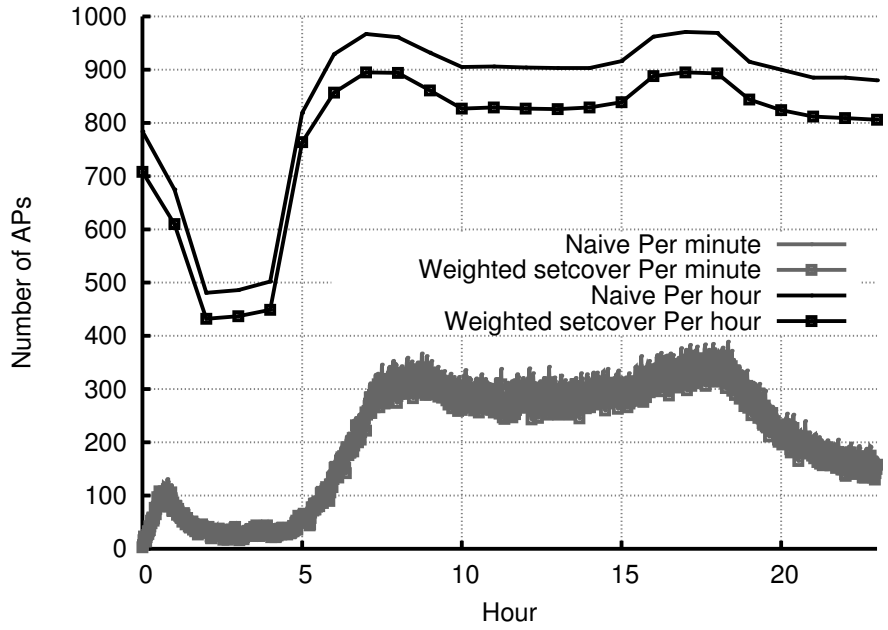


Figure 5.33: Total number of unique APs used in DDoS per unit time during weekdays by bus botnet

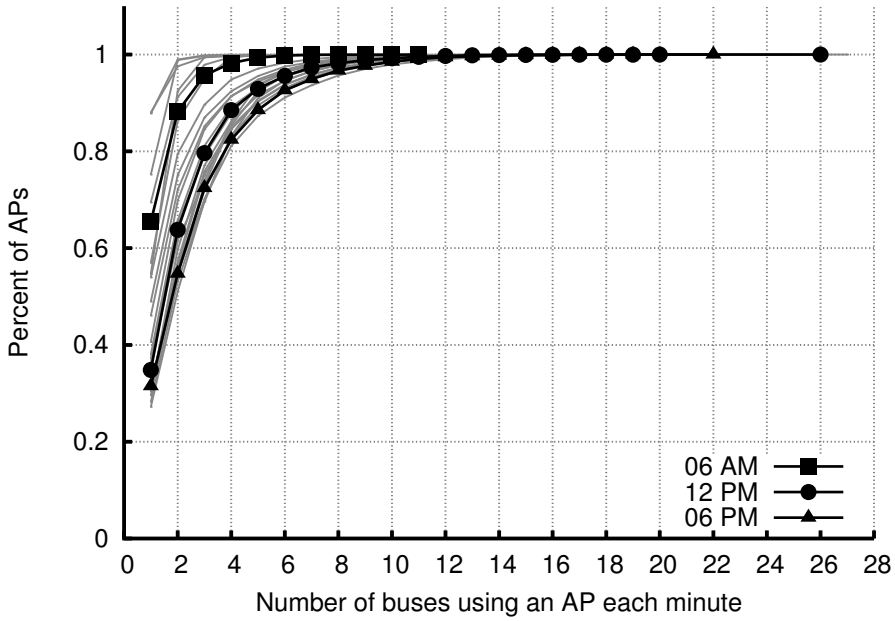


Figure 5.34: CDF of number of bots performing DDoS at each AP per minute for Weighted Setcover during weekdays for bus botnet

5.5.4 Spam Attack

In this section, we aim to answer the following questions concerning mobile WiFi botnet spam attacks:

- What is the botnet's capacity for spam attacks?
- How are spam attacks distributed across open APs?

For the cab botnet, we plot the lower- and upper-bound (i.e., spam_lower and spam_upper) average number of spam emails sent hourly during weekday rush hours, Saturdays, and Sundays in Figs. 5.35–5.38. From the figures, we find that the amount of spam sent follows the familiar diurnal patterns seen in the previous, less data-intensive, C&C and DDoS attacks. Furthermore, between the lower- and upper-bounds, there is an order-of-magnitude disparity in the quantity of spam sent, with spam_lower being capable of issuing tens of thousands of spam emails hourly compared to spam_upper's hundreds of thousands. For the weekday rush hours, both achieve their best results during the second half of the morning and afternoon commute hours (i.e., 8 and 9 a.m. and 5:30 and 6:30 p.m.). During these times, spam_lower can typically issue over 20,000 spam emails hourly, while spam_upper can typically issue over 300,000; the exception being Mondays, when slightly fewer emails can be sent. During the other weekday commute hours, spam_lower and spam_upper can, on average, send over 15,000 and 200,000 spam emails, respectively. Even amidst their weakest hour of 6 a.m., spam_lower can still issue over 10,000 emails an hour and spam_upper over 150,000. Despite utilizing only these 8 weekday rush hours, this amounts to a considerable volume of daily spam, as shown in Figs 5.39 and 5.41 for spam_lower and spam_upper, respectively. Independent of the day, spam_lower can send between ≈130,000 and 160,000 spam emails, while spam_upper can attain the considerably higher range of ≈1.9 to 2.3 million. For both spam_upper and spam_lower, we observe a similar allotment of spam sent hourly during the weekends

as is seen during the weekday rush hours. However, the magnitude sent follows a different diurnal trend due to the weekends possessing mobility patterns distinct from the weekdays. For instance, the optimal weekday hours of 8 and 9 a.m. and 5:30 and 6:30 p.m. are not the optimal hours during the weekends. On the contrary, Sunday's spam throughput is fairly consistent from 7 a.m. until midnight, with only minor oscillations, while Saturday's peak hours are observed during the evenings, when people are likely out to dinner and/or the bars; in fact, Sunday's maximum spam throughput occurs between midnight and 1 a.m., when people are returning from a Saturday night out. Throughout Saturday's evening hours (i.e., from 6 p.m. to 1 a.m.), spam_lower is able to issue over 20,000 emails hourly, while spam_upper is able to issue over 300,000. For both Saturdays and Sundays, spam_lower rarely falls below 15,000 emails per hour, and spam_upper rarely falls below 200,000. Even during their worst hours, both spam_lower and spam_upper can issue a formidable amount of spam; during the early morning hours of 4 and 5 a.m., spam_lower can send over 6,000 emails, while spam_upper can send over 100,000. The spamming potential during the weekends becomes even more apparent when viewed as a daily aggregate, with spam_lower able to issue between ≈400,000 and 450,000 spam emails daily and spam_upper between ≈5.5 and 6.5 million, as shown in Figs. 5.40 and 5.42, respectively. The sizable gulf that exists between the upper- and lower-bound spamming capabilities evinces the potential for improvement that can be made with more intelligent spamming methods. However, even the lower-bound quantity of spam sent is daunting, numbering in the hundreds of thousands daily and originating from only a small mobile botnet, which never exceeds 536 simultaneously active bots. Should our highly mobile botnet prove as adept in spreading its spamming traffic across numerous APs as it has with its other attack traffic, it will further attest to the severity of the future mobile botnet threat lurking on the horizon.

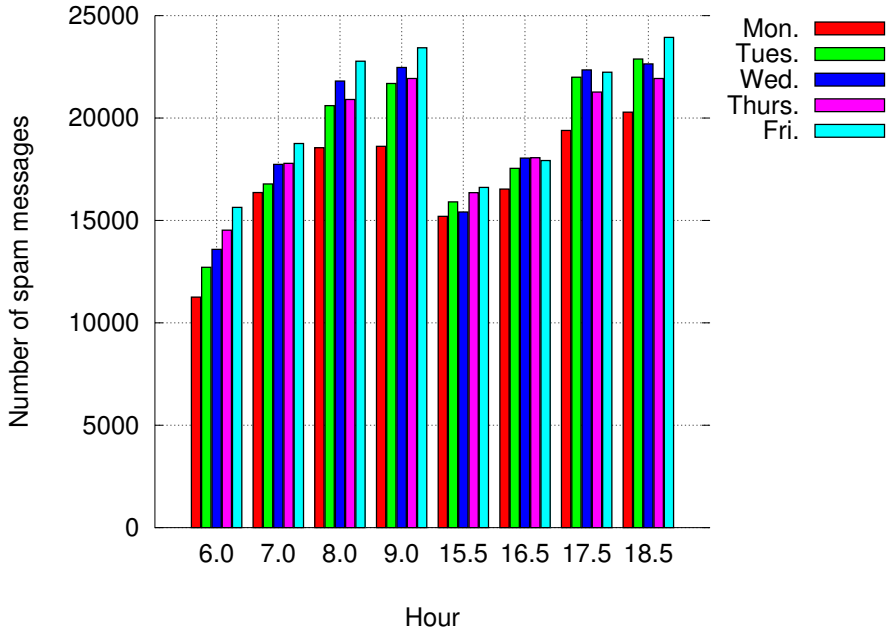


Figure 5.35: Average number of spam messages sent each hour during weekday rush hours for spam_lower by cab botnet

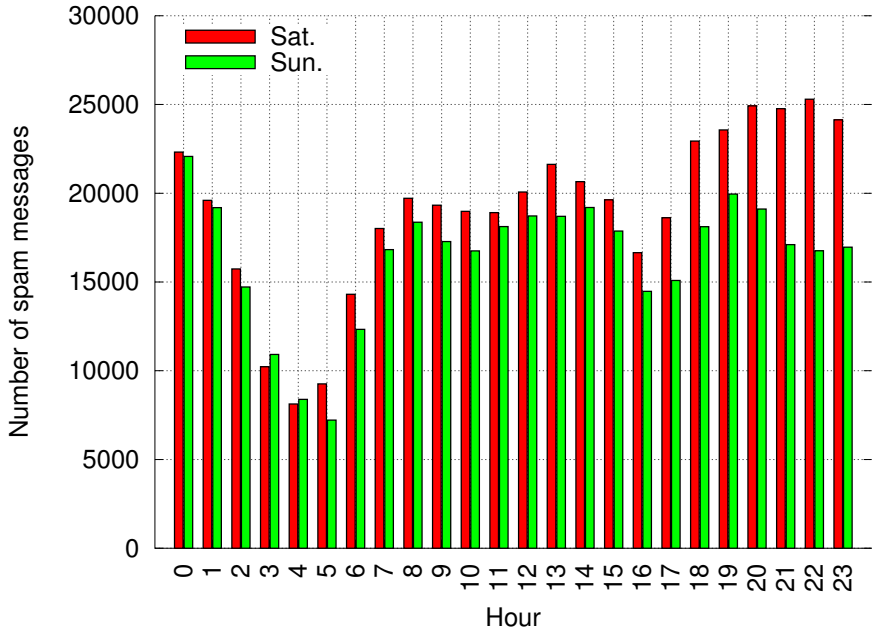


Figure 5.36: Average number of spam messages sent each hour during weekends for spam_lower by cab botnet

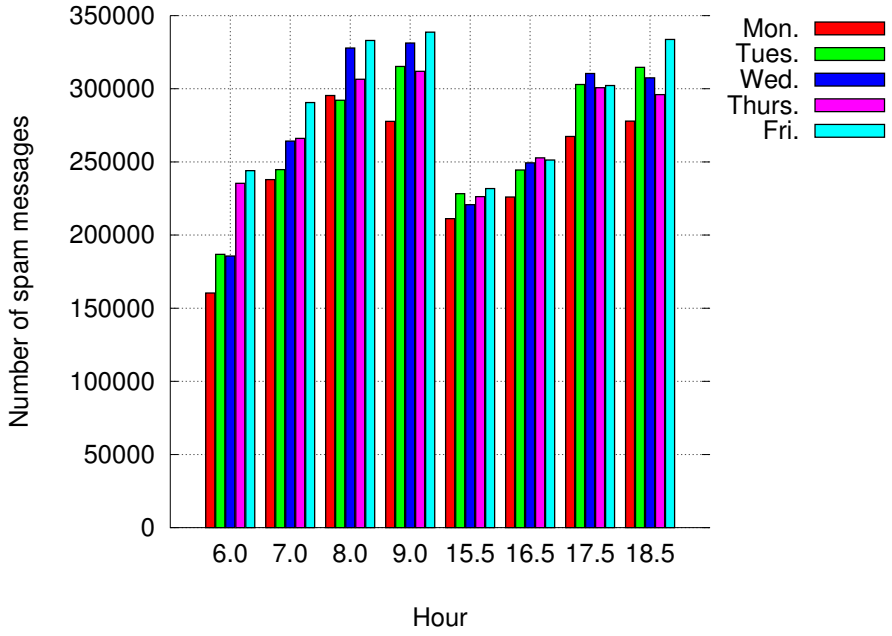


Figure 5.37: Average number of spam messages sent each hour during weekday rush hours for spam_upper by cab botnet

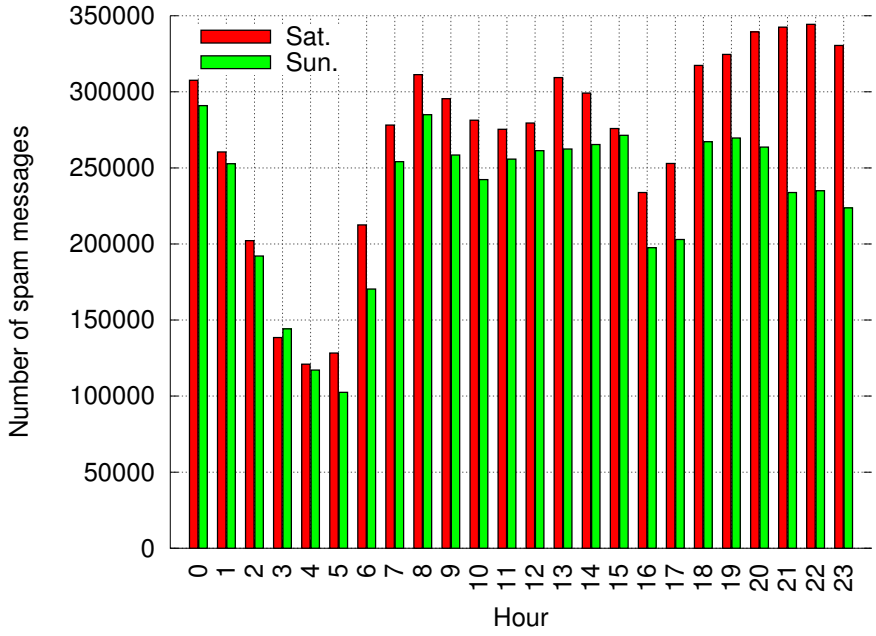


Figure 5.38: Average number of spam messages sent each hour during weekends for spam_upper by spam cab botnet

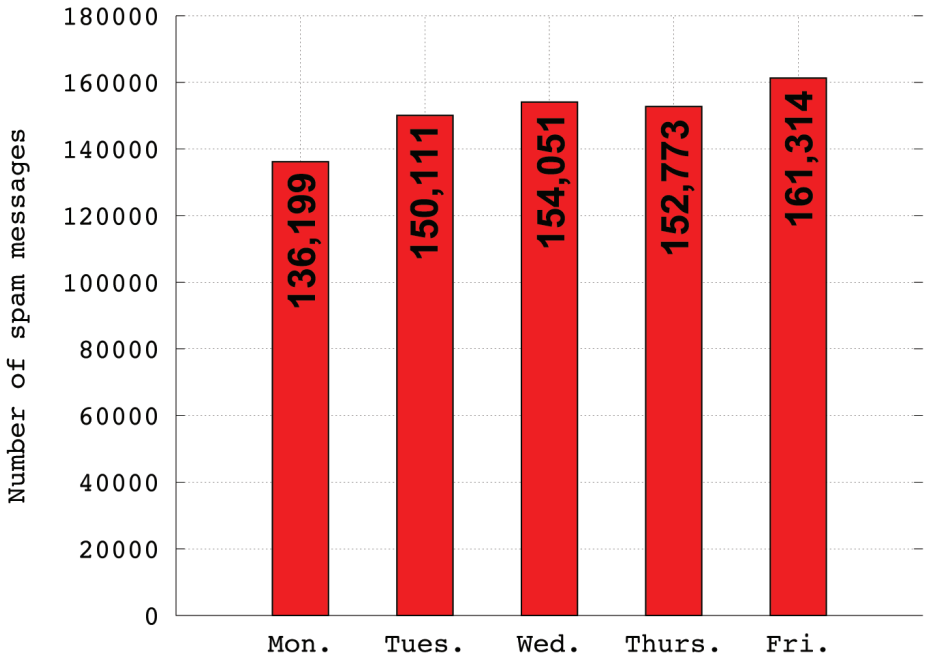


Figure 5.39: Average total number of spam emails sent per day during weekday rush hours for spam_lower by cab botnet

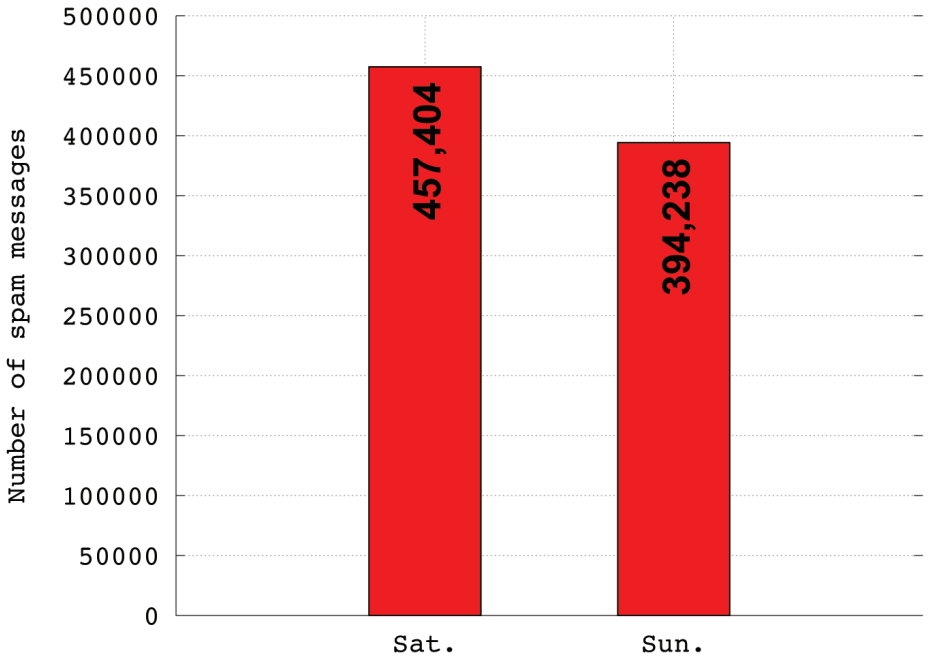


Figure 5.40: Average total number of spam emails sent per day during weekends for spam_lower by cab botnet

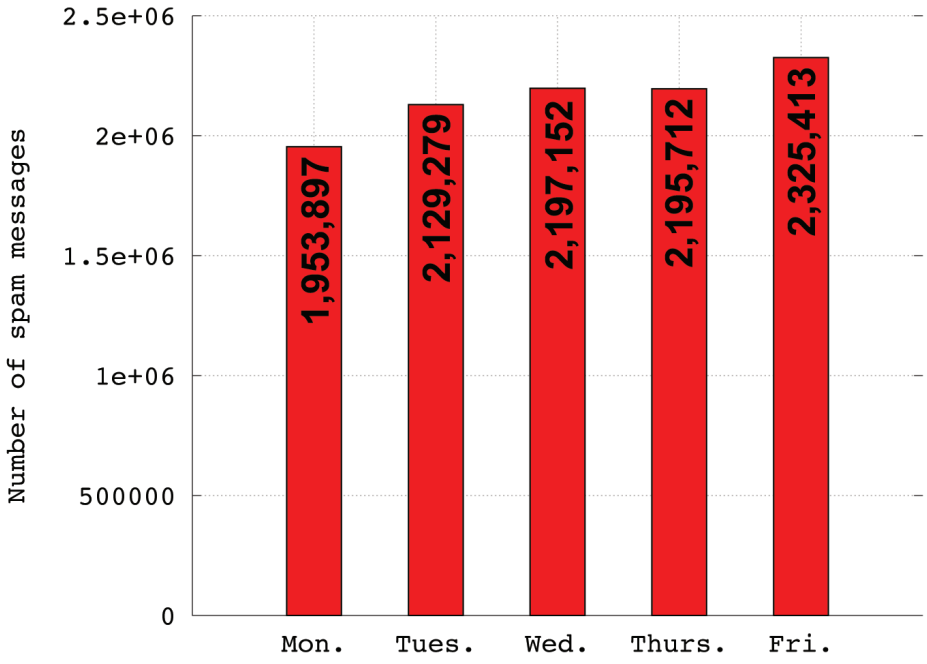


Figure 5.41: Average total number of spam emails sent per day during weekday rush hours for spam_upper by cab botnet

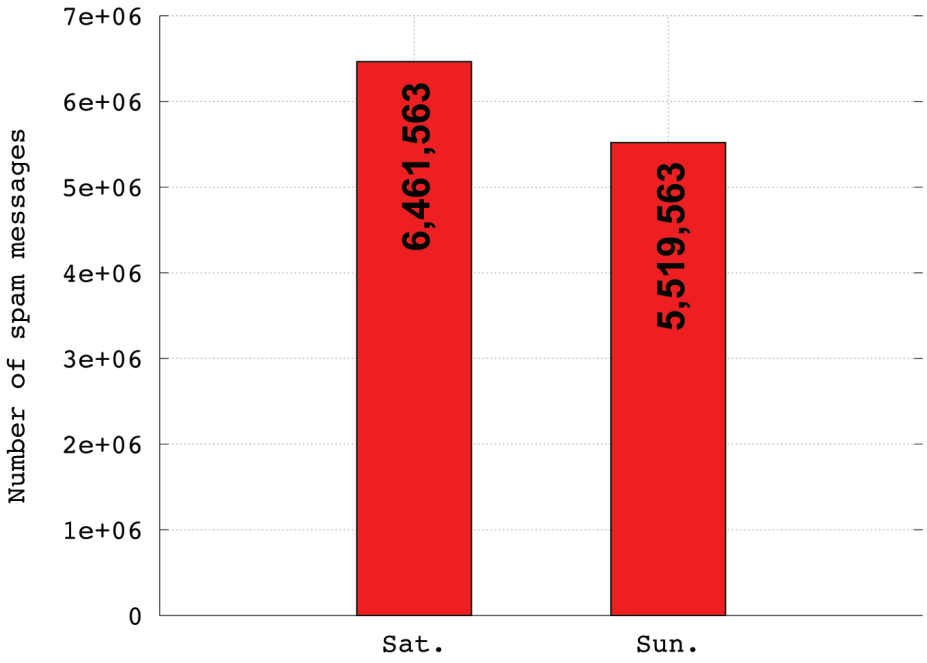


Figure 5.42: Average total number of spam emails sent per day during weekends for spam_upper by cab botnet

For the bus botnet's various AP-selection algorithms, the amount of lower- and upper-bound spam emails sent during weekdays and Sunday are shown in in Figs 5.43–5.46; Saturday has been omitted due to its similarity to Sunday. For both spam_lower and spam_upper, Setcover and Setcover patched performed better than the Naïve algorithms but only slightly worse than the Weighted Setcover variants. For example, for spam_lower during the peak hour of 5 p.m. on weekdays, Setcover patched issued ≈1,000 fewer (≈98.7%) spam emails than Weighted Setcover and its patched version; likewise, Setcover issued ≈1,100 fewer (≈98.5%) than Setcover patched, and Naïve issued ≈1,200 fewer (≈98.1%) than Naïve patched. More significant gains could be found between Naïve patched and Setcover, with Naïve patched issuing ≈7,900 fewer (≈89.4%) spam emails hourly than Setcover; this is ≈7–8x the improvement observed between any other pair of AP-selection algorithms. The improvement between the simple Naïve and the more advanced Weighted Setcover[10] is even more pronounced, with Naïve issuing ≈65,700 emails at 5 p.m. on weekdays compared to Weighted Setcover's ≈77,000—over a 17% increase in spam emails sent. Saturday and Sunday have similar results, though the quantity of spam is slightly less as there are fewer buses in service on weekends than the weekdays—e.g., at 5 p.m. on Sundays, ≈15,000 fewer spam emails are sent per AP-selection algorithm than on weekdays. Even during the early morning hours, when there is less distinction between the various AP-selection algorithm's output, they are all still capable of spamming ≈2,000 targets hourly on both weekdays and weekends.

The intelligent AP-selection algorithms produce similar results for spam_upper, though the sheer magnitude in the amount of spam sent is greatly increased. For example, for spam_upper during the peak hour of 5 p.m. on weekdays, Setcover patched issued ≈8,500 fewer (≈99.2%) spam emails than Weighted Setcover and its patched version; likewise, Setcover issued ≈5,200 fewer (≈99.5%) than Setcover patched, and

[10]Recall, Weighted Setcover and its patched variant produced nearly identical results.

Naïve issued ≈5,600 fewer (≈99.4%) than Naïve patched. As with spam_lower, the most significant gains were found between Naïve patched and Setcover, with Naïve patched issuing ≈95,900 fewer (≈90.7%) spam emails than Setcover—between ≈11–18x the improvement observed between any other pair of AP-selection algorithms. Once again, the greatest improvement is observed between Naïve and Weighted Setcover; however, spam_upper's use of multiple recipients has drastically increased the amount of spam sent. For example, at 5 p.m., Naïve can issue ≈934,000 spam on weekdays and ≈738,000 on Sundays. The use of Weighted Setcover has increased this quantity by over 12%, to ≈1,050,000 on weekdays and ≈830,000 on Sunday.

These similar results for spam_lower and spam_upper clearly demonstrate the room for improvement that intelligent AP-selection can provide, accentuating the benefits of more advanced approaches over simpler ones based on heuristics, such as our patch-based solutions. While simple, heuristic approaches are easier to implement and require less a priori knowledge, they cannot produce the level of improvement attainable through more complex techniques assimilating various sources of data into their solution. More intelligent AP-selection methods than our Setcover and Weighted Setcover variants could provide botmasters with even greater improvements, further increasing the threat of mobile botnet spam. Furthermore, the figures demonstrate the type of gains that can be achieved from issuing multiple spam emails to a single mail server, with spam_upper able to send an order-of-magnitude (i.e, ≈12–15x) more spam than spam_lower, independent of the day or AP-selection algorithm. These dramatic increases are also observed during the early morning hours, when the number of buses in service is at its minimum. At these times, spam_upper can send ≈23,000–39,000 emails on weekdays and ≈19,000–30,000 on Sunday—between 25–50% of the maximum amount spam_lower can issue during its optimal hour. The discrepancy between these two spam protocols can be further witnessed in Fig. 5.47, which shows the total amount of spam sent daily by each protocol for Naïve and Weighted Set-

cover. While our spam_upper protocol represents a theoretical maximum unlikely to occur in nature, it demonstrates that using multiple recipients for common email domains can result in improvements over the simpler one-to-one approach—a technique made practical by the popularity and prevalence of free email services, such as GMail, Hotmail, and Yahoo! mail.

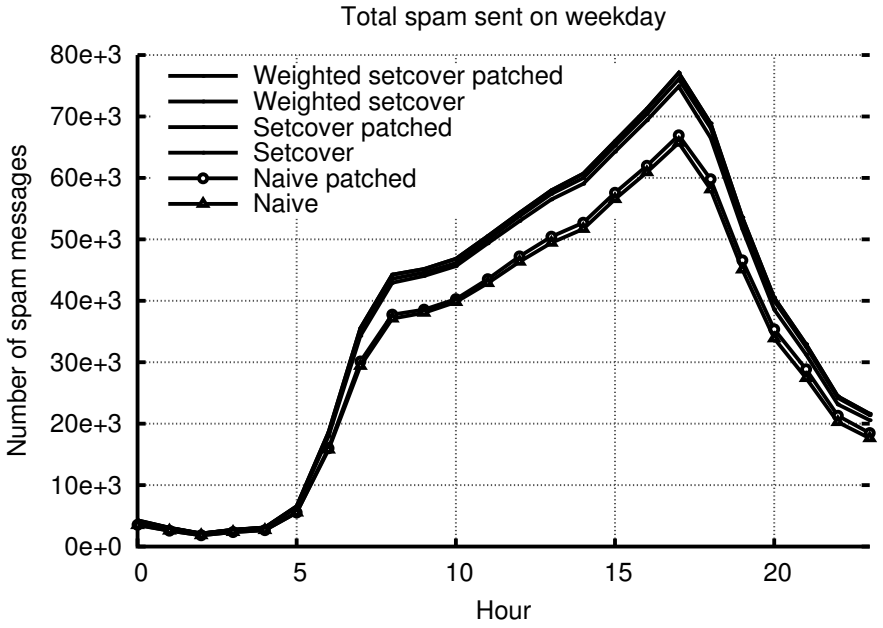


Figure 5.43: Number of spam messages sent each hour during weekdays for spam_lower by bus botnet

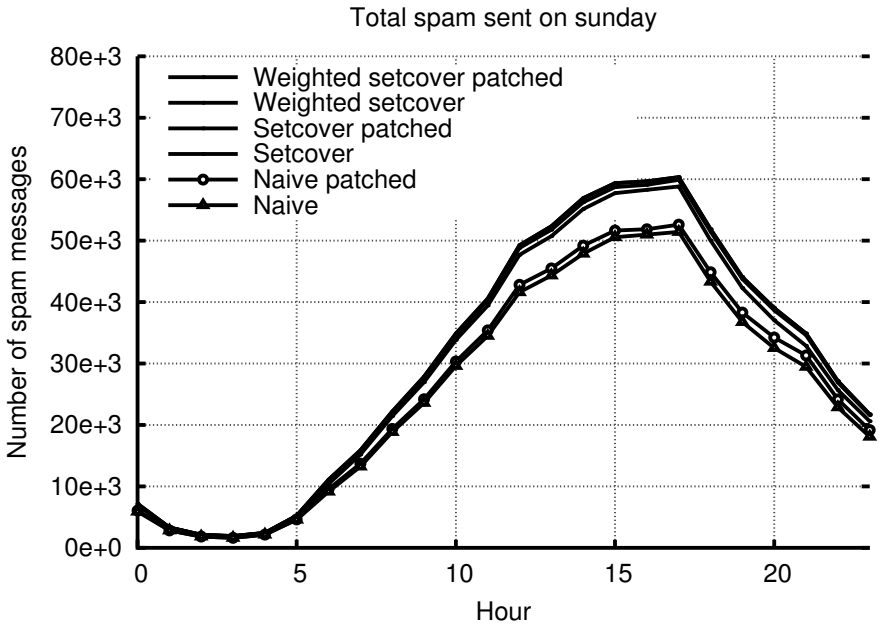


Figure 5.44: Number of spam messages sent each hour on Sunday for spam_lower by bus botnet

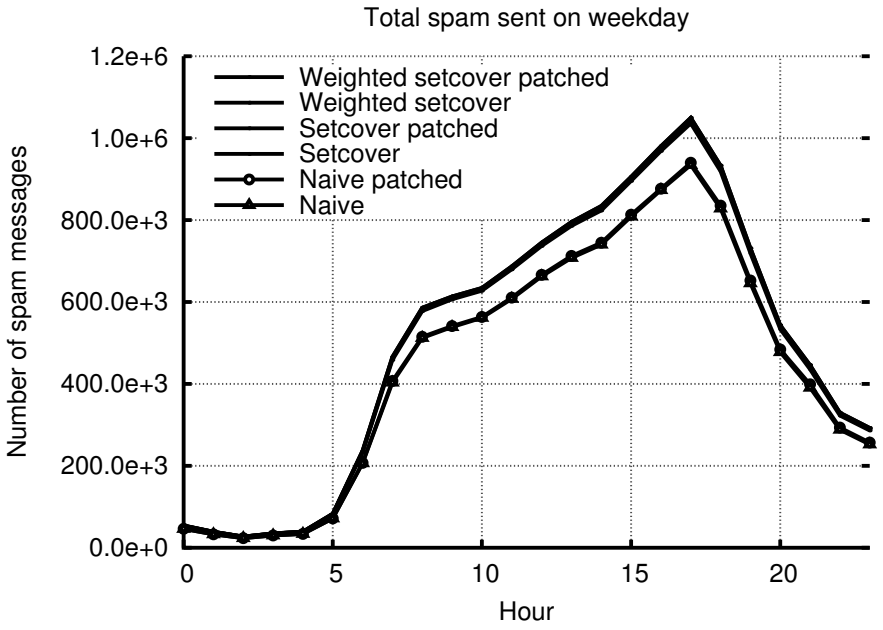


Figure 5.45: Number of spam messages sent each hour during weekdays for spam_upper by bus botnet

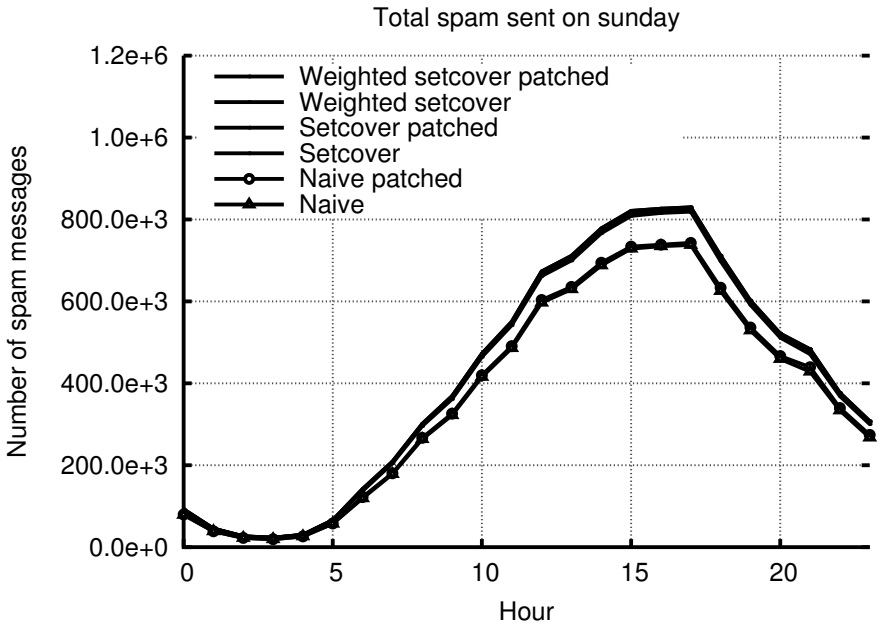


Figure 5.46: Number of spam messages sent each hour on Sunday for spam_upper by spam bus botnet

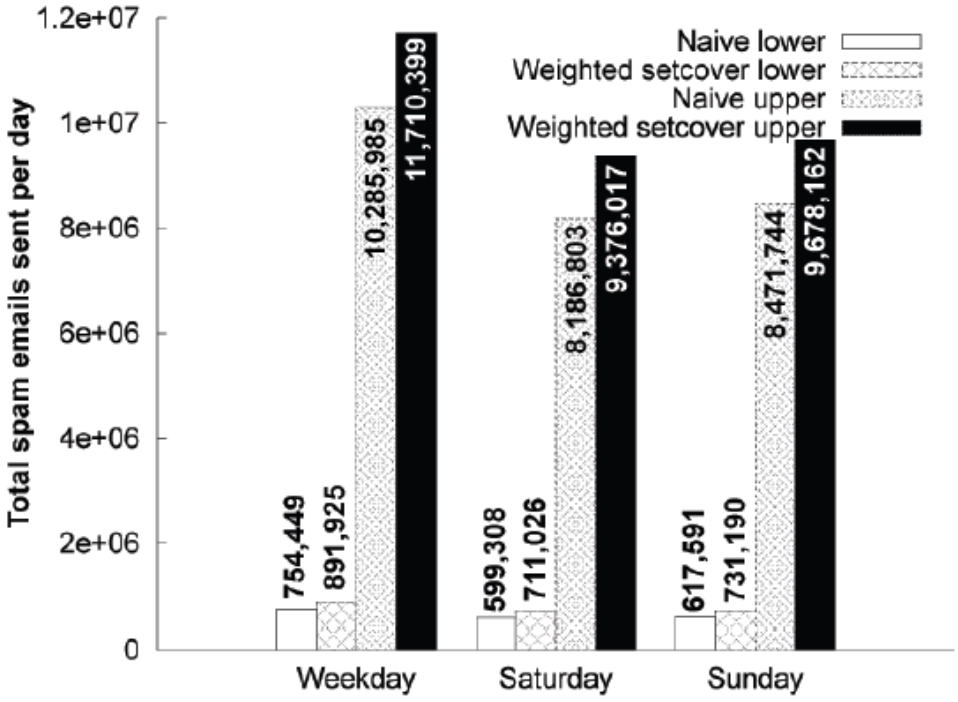


Figure 5.47: Total number of spam emails sent per day by bus botnet

As previously done for C&C and DDoS, we next examine if the cab and bus botnet spam attacks are sufficiently distributed across APs to encumber detection. We first examine the cab botnet, plotting the number of bots sending spam at each AP per minute during weekday rush hours, Saturdays, and Sundays for spam_lower and spam_upper in Figs. 5.48–5.53. Additionally, we plot the maximum and average number of APs used during each minute and hour by the cab botnet during weekday rush hours, Saturdays, and Sundays for both spam_lower and spam_upper in Figs. 5.54–5.59. These results, which are similar for both spam_lower and spam_upper independent of the day, indicate that the cab botnet's spam attack is well distributed across open APs, complicating detection. For example, for any given day and hour, for both spam_lower and spam_upper, ≈80–94% of open APs are utilized by only a single bot per minute, and ≈99.1–99.6% of open APs are used by 3 or fewer bots per minute. Of the ≈0.9–0.4% of open APs that do service more than 3 bots per minute, none ever exceed 10–15 bots per minute. While 10–15 bots issuing spam behind a single open AP could simplify detection, it should be noted that these extremes only occur for a very limited number of APs, and when they do occur, it is only for a minute or less before the bots migrate to another open AP; even if the bots are discovered during this short window, they will have soon transferred to another open AP where they can continue their attack unabated.

Figures 5.54–5.59 affirm the distributed nature of the cab botnet's spam attack. With the exception of the early morning hours, the botnet typically makes use of more than ≈1,000 unique open APs per hour, or ≈100–200 per minute. Even during the early morning hours, when there are considerably fewer mobile bots in service, the botnet still spreads its attack over a significant number of open APs. For example, during 6 a.m. on weekdays, it makes use of over ≈900 distinct APs per hour, while during the extreme weekend lull from 3–6 a.m., it spreads the attack across ≈700–900 APs per hour. Also from these figures, we can see that the average number of unique

APs used, both hourly and per minute, is nearly as many as the maximum, indicating that the attack is consistently well distributed throughout the day and not oscillating over a wide and unpredictable range.

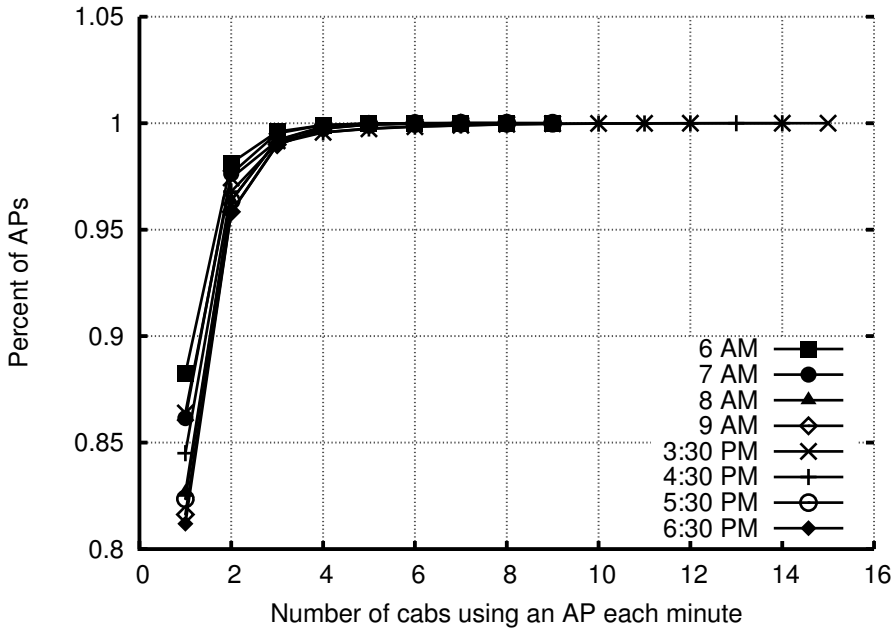


Figure 5.48: CDF of number of bots sending spam at each AP per minute for spam_lower during weekday rush hours for cab botnet

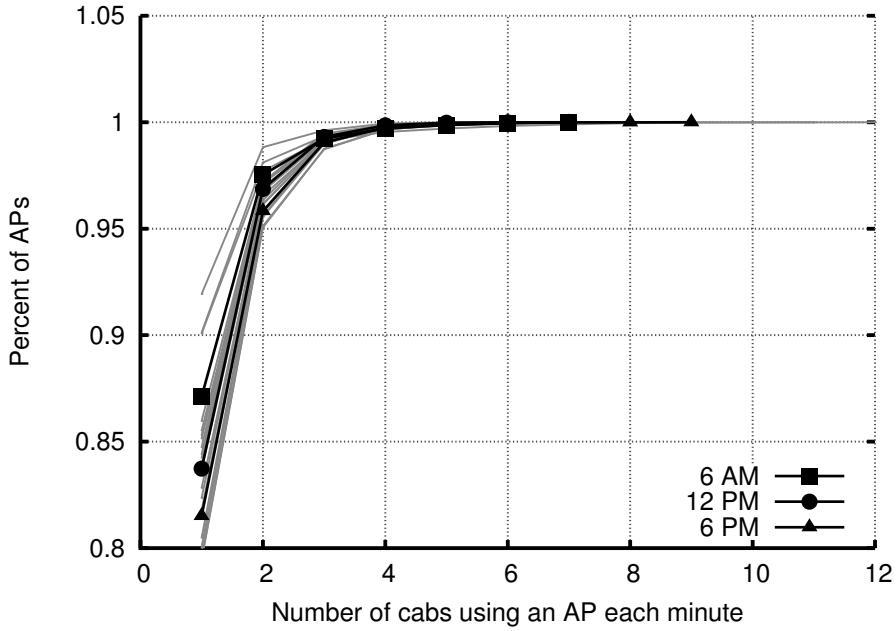


Figure 5.49: CDF of number of bots sending spam at each AP per minute for spam_lower on Saturday for cab botnet

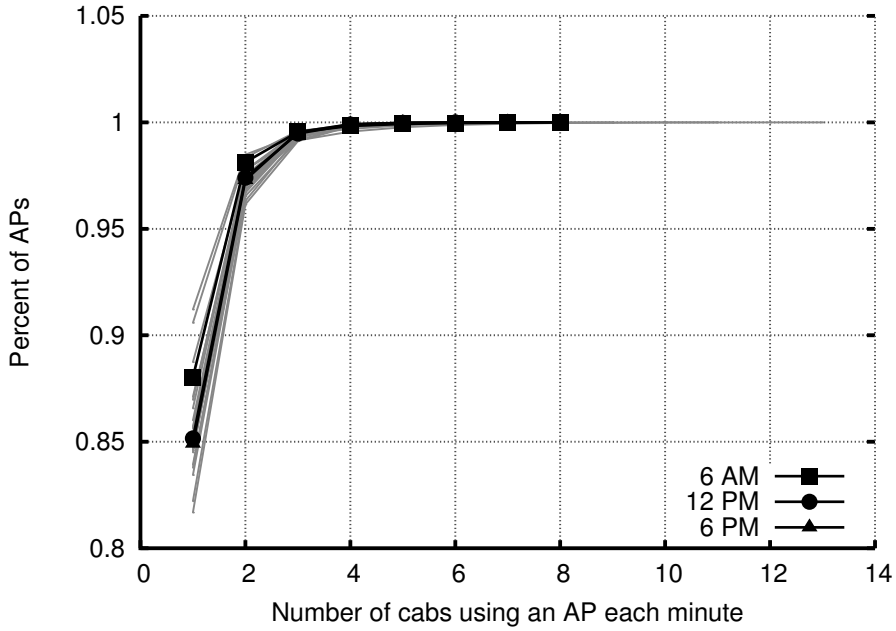


Figure 5.50: CDF of number of bots sending spam at each AP per minute for spam_lower on Sunday for cab botnet

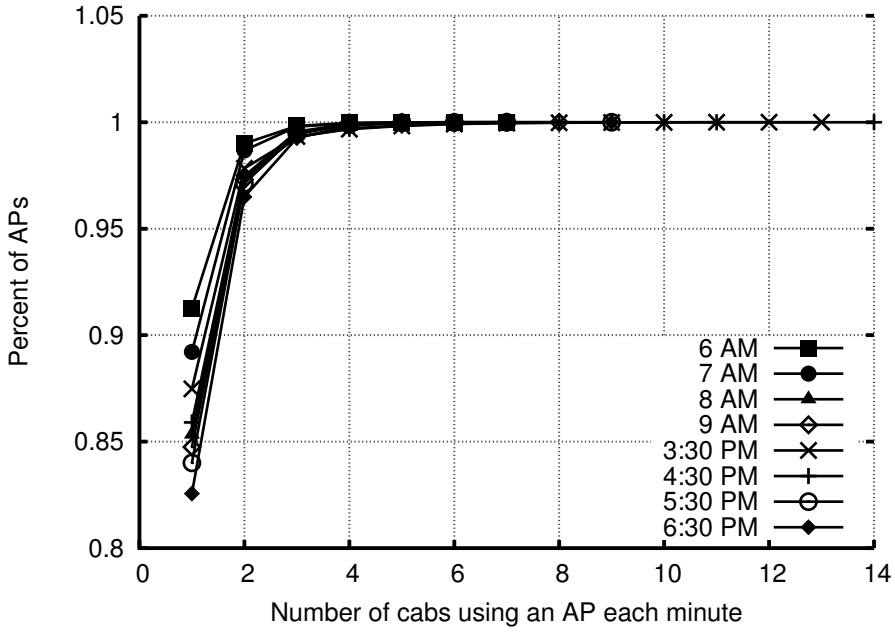


Figure 5.51: CDF of number of bots sending spam at each AP per minute for spam_upper during weekday rush hours for cab botnet

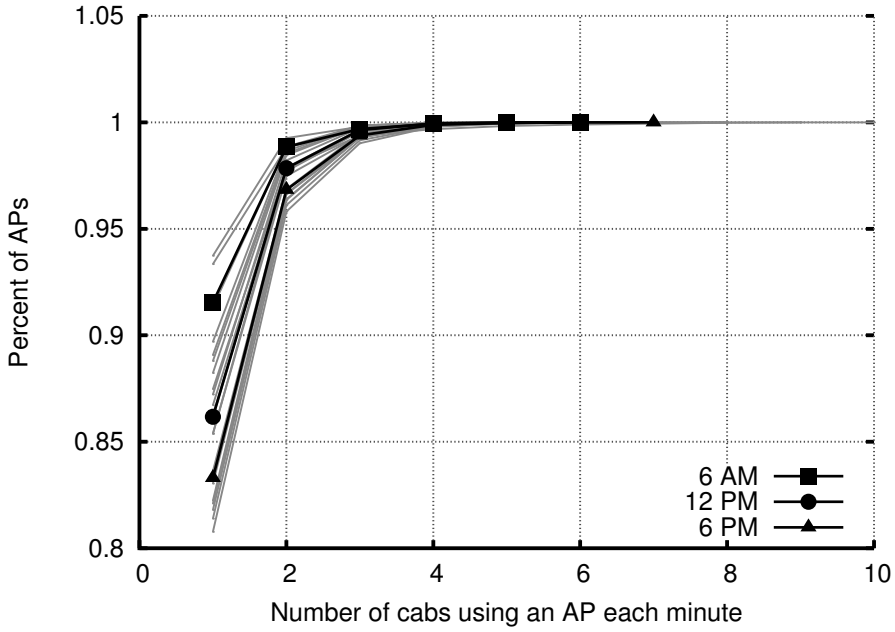


Figure 5.52: CDF of number of bots sending spam at each AP per minute for spam_upper on Saturday for cab botnet

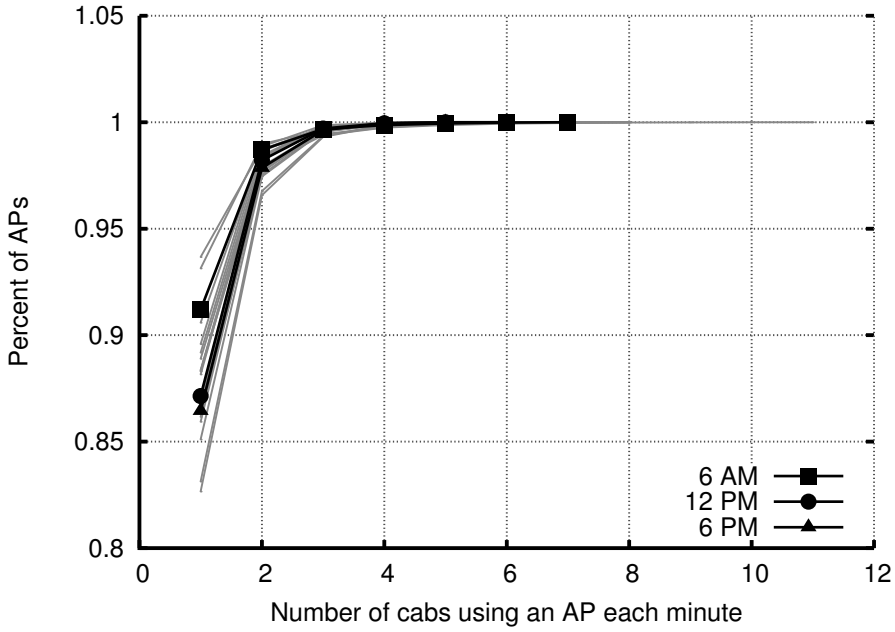


Figure 5.53: CDF of number of bots sending spam at each AP per minute for spam_upper on Sunday for cab botnet

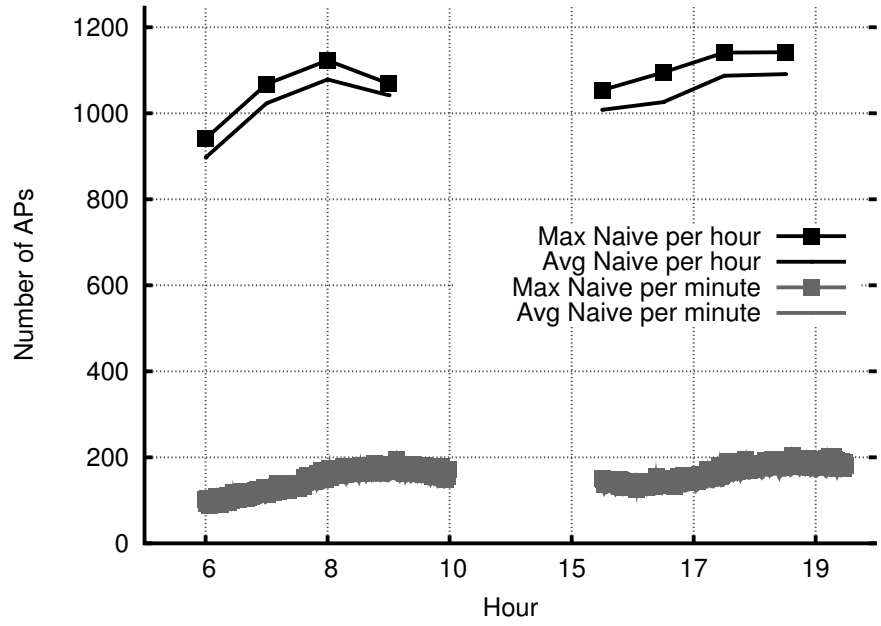


Figure 5.54: Average and max number of unique APs used for spam_lower per unit time during weekday rush hours by cab botnet

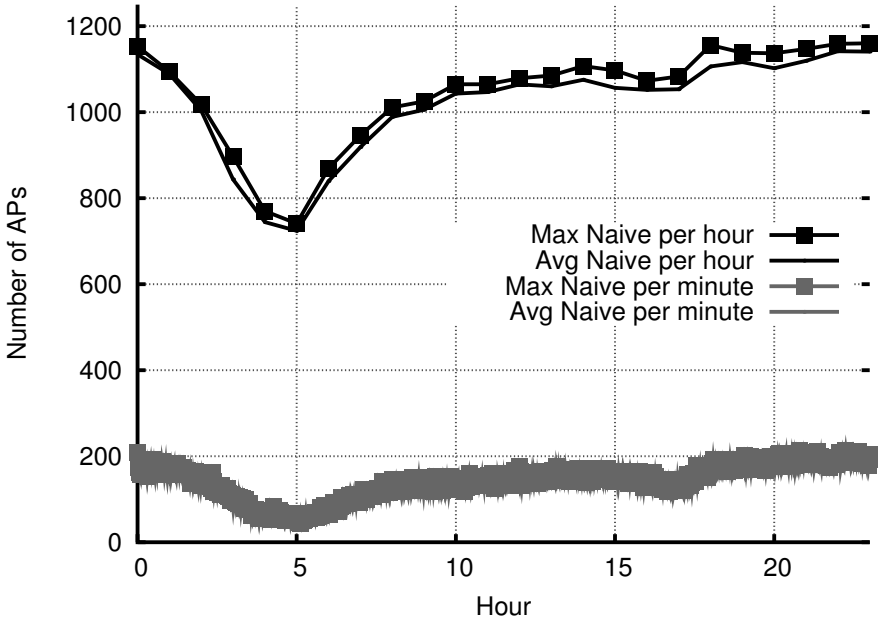


Figure 5.55: Average and max number of unique APs used for spam_lower per unit time on Saturday by cab botnet

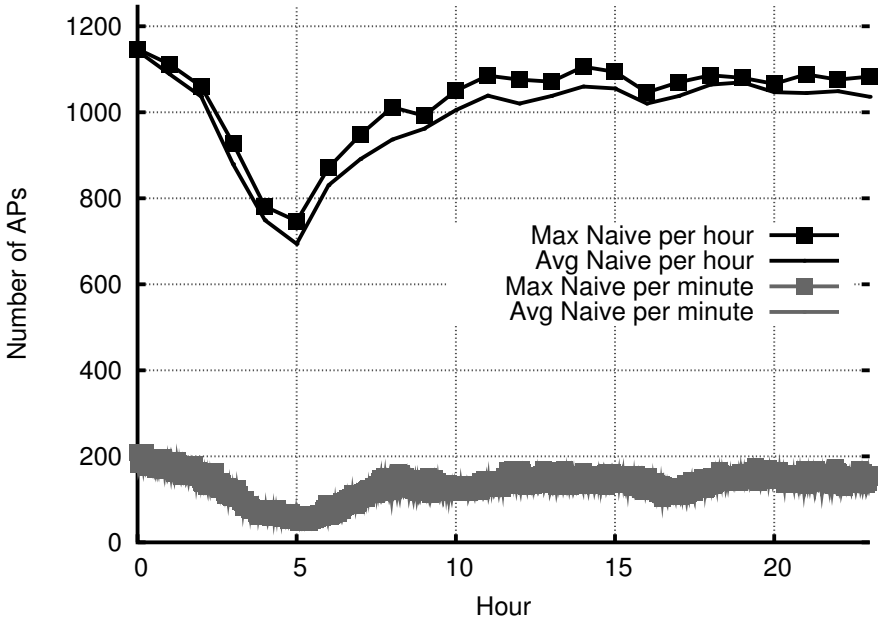


Figure 5.56: Average and max number of unique APs used for spam_lower per unit time on Sunday by cab botnet

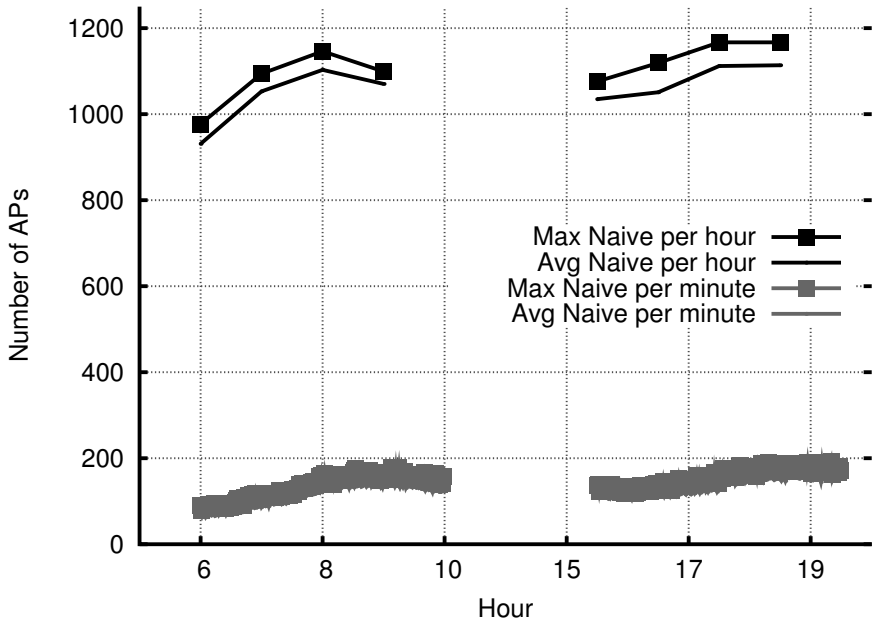


Figure 5.57: Average and max number of unique APs used for spam_upper per unit time during weekday rush hours by cab botnet

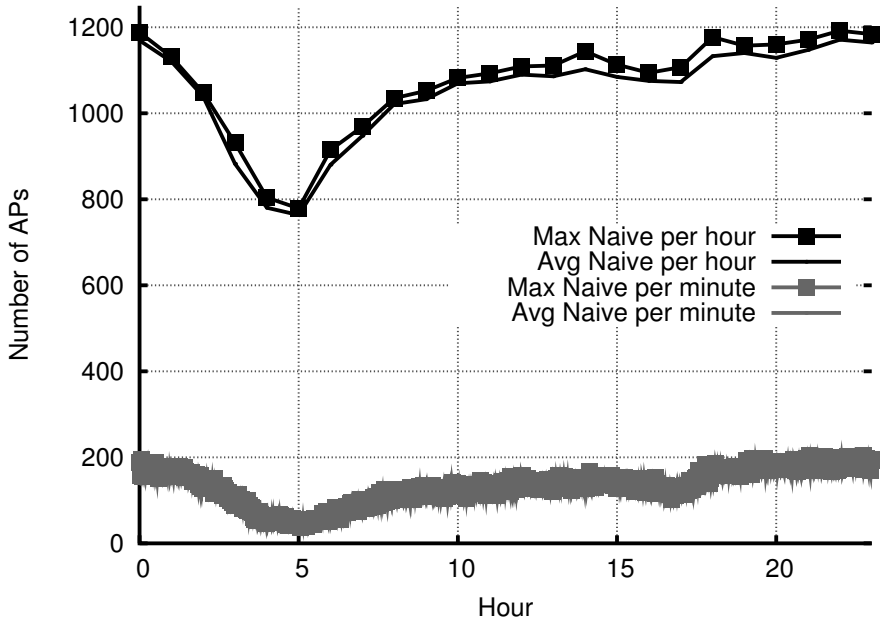


Figure 5.58: Average and max number of unique APs used for spam_upper per unit time on Saturday by cab botnet

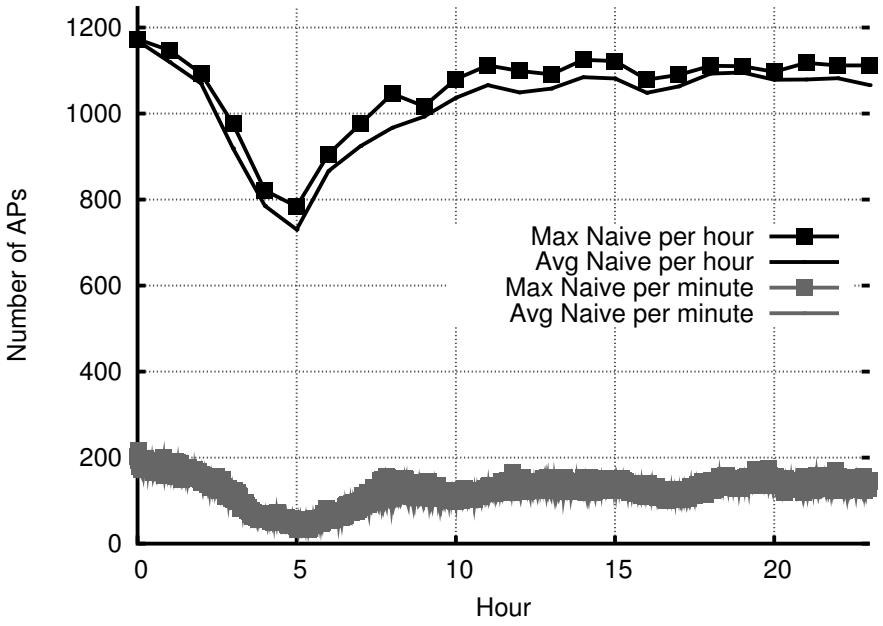


Figure 5.59: Average and max number of unique APs used for spam_upper per unit time on Sunday by cab botnet

For the bus botnet, the number of bots sending spam at each AP per minute when using Weighted Setcover during weekdays is shown in Figs 5.60 and 5.61 for spam_lower and spam_upper, respectively; the results were similar for Saturday and Sunday as well as the other AP-selection algorithms. Figures 5.62 and 5.63 show spam_lower's and spam_upper's total number of APs used per minute and hour by the Naïve and Weighted Setcover AP-selection algorithms; the other AP-selection algorithms fell between the bounds of the two end-cases displayed, with Saturday and Sunday producing similar results. From the figures, we find that, as with the cab botnet, both spam_lower and spam_upper are sufficiently distributed across numerous APs. Furthermore, for all hours in both attacks, ≈90% of the APs are used by 5 or fewer simultaneous bots per minute, and ≈99% are used by 8 or fewer. Lastly, we notice that because Weighted Setcover attempts to minimize AP switches, it naturally uses fewer unique APs per hour than Naïve; even so, it still makes use of hundreds of unique APs per hour and is sufficiently distributed to encumber detection.

Because spam emails issued by a mobile WiFi botnet originate from multiple APs, with bots sending a few at each network before switching, detection and mitigation of the spam sources becomes incredibly difficult. Considering that the amount of spam sent by our lower-bound is in the hundreds of thousands daily, mobile WiFi botnets should be considered a serious potential threat as a discrete and powerful spamming mechanism for botmasters.

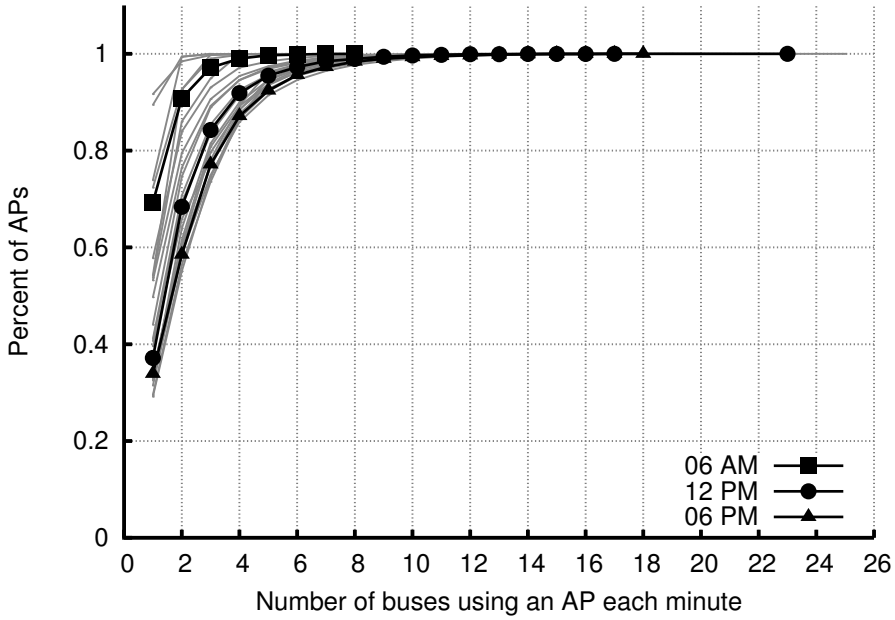


Figure 5.60: CDF of number of bots sending spam at each AP per minute for spam_lower using Weighted Setcover during weekdays for bus botnet

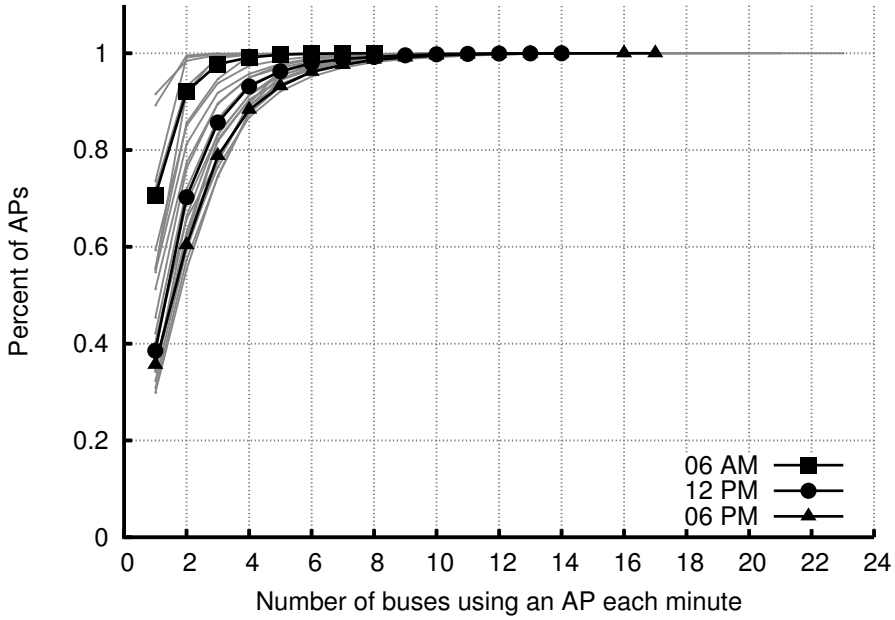


Figure 5.61: CDF of number of bots sending spam at each AP per minute for spam_upper using Weighted Setcover during weekdays by bus botnet

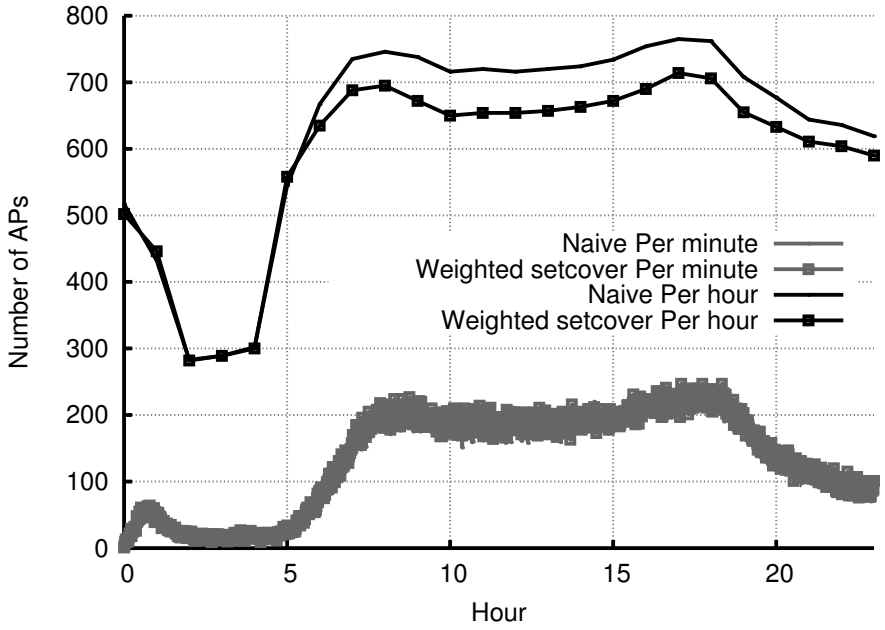


Figure 5.62: Total number of unique APs used for spam_lower per unit time during weekdays by bus botnet

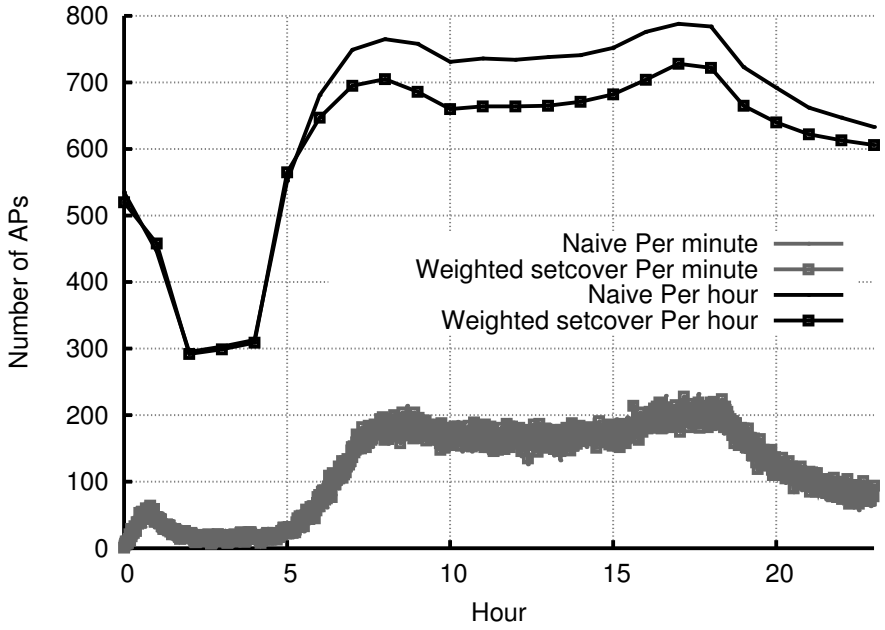


Figure 5.63: Total number of unique APs used for spam_upper per unit time during weekdays by bus botnet

5.6 Defense Strategies

The most obvious defense strategy is to require WiFi APs to use some form of encryption, such as WEP or WPA2, reducing the number of open APs available to mobile botnets and diminishing their overall capabilities. However, this approach may not be suitable for many WiFi networks, whose users may desire or require the networks to be unencrypted (e.g., restaurant and cafes, see Section 5.3.1). An alternate strategy can be employed that imposes an association delay on open WiFi APs, making users wait a certain amount of time before granting online access.[11] Depending on the delay's duration, mobile bots traveling in vehicles may not be within range of APs long enough to complete the C&C protocol.[12] Likewise, a delay limits the number of SYN packets and spam emails mobile bots can issue from a given AP.

We simulate this strategy for several different delay periods. For weekday rush hours and various delays, Figs. 5.64 and 5.65 show the average number of cabs able to receive commands and the average cab responsiveness. While even modest delays of 5 to 10 seconds can impact the average cab responsiveness, the number of cabs able to receive commands is not significantly diminished until larger delays of 30 and 60 seconds are enforced. This is reaffirmed in Fig. 5.66, which shows the average propagation rate for different delay periods when a command is issued at 9 a.m. on a weekday. Only with 30- and 60-second delays do we begin to see a significant decrease in propagation. Longer delays degrade the botnet's C&C even further, with a 5-minute delay reducing command propagation from ≈80% after 2 hours down to only ≈20%, and 10-minute delay to less than 10%.

[11] Notice, open WiFi networks that require users to click an "Accept" button before granting access are essentially implementing a very short delay, as bots can automatically "click" the button to gain access.

[12] Clearly, this approach is best suited for highly mobile bots, such as those traveling in vehicles, and it is unlikely to achieve as good as results with the slower mobility of pedestrians.

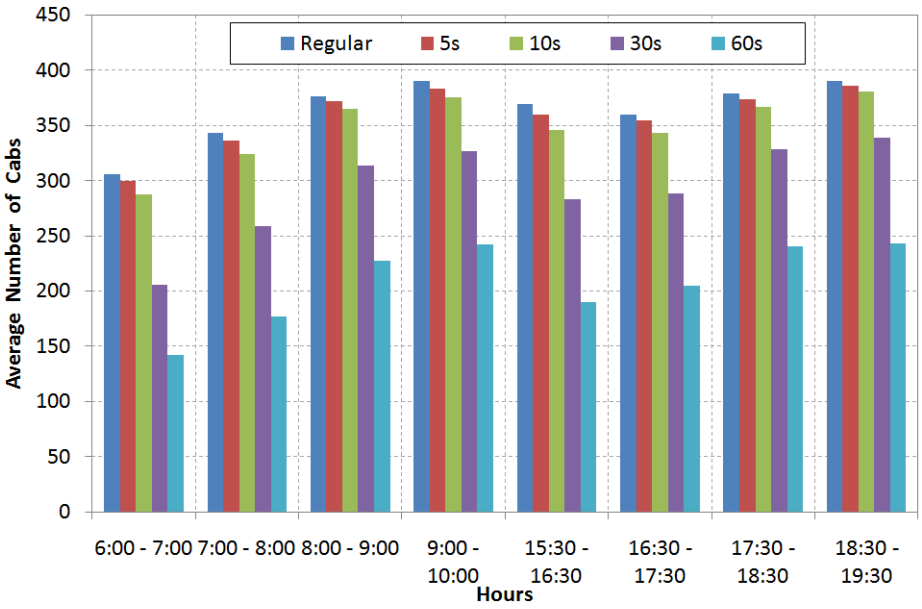


Figure 5.64: Average number of bots receiving commands during weekday rush hours for various delays for cab botnet

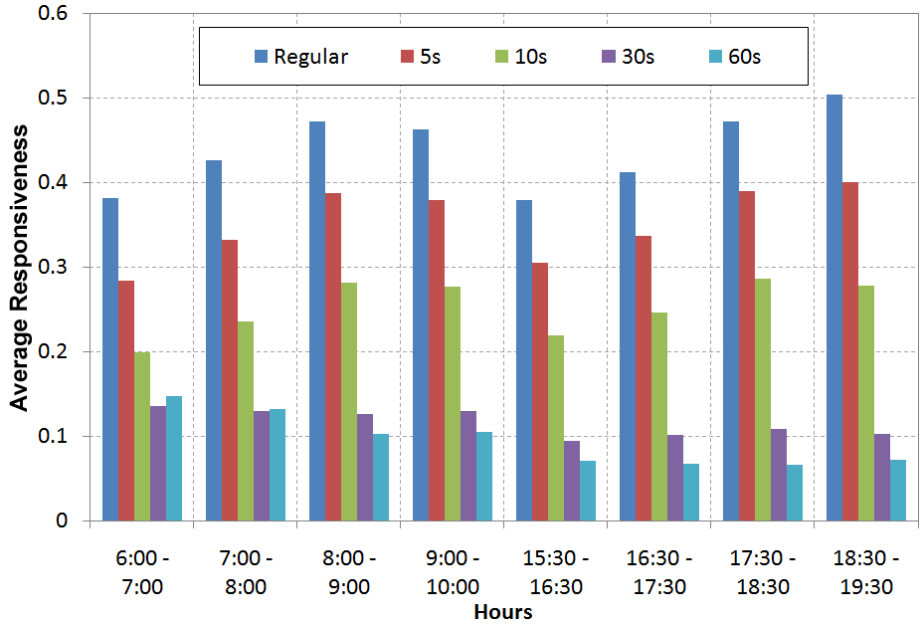


Figure 5.65: Average bot responsiveness during weekday rush hours for various delays for cab botnet

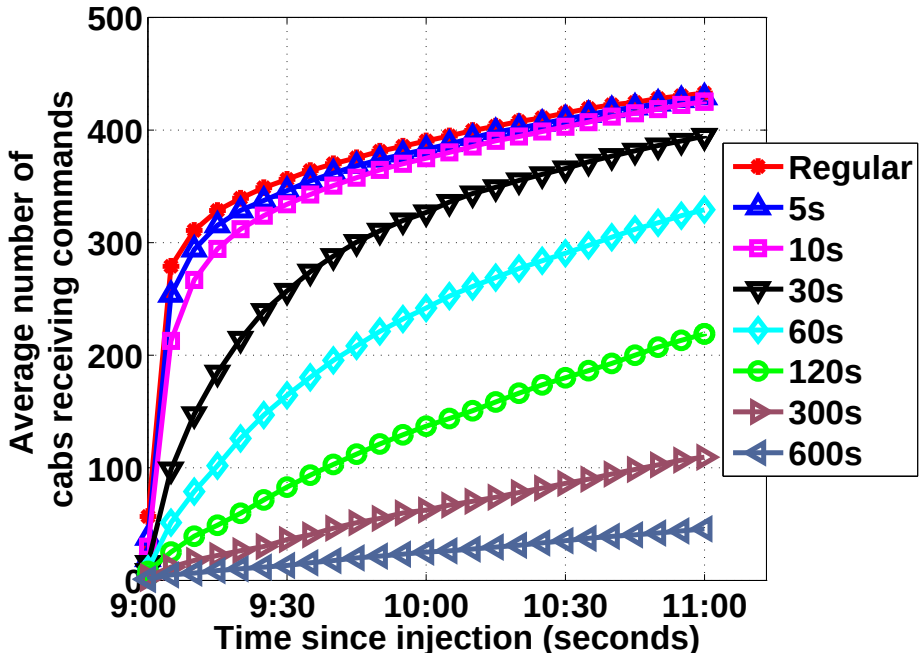


Figure 5.66: Average number bots receiving commands after a 9 a.m. injection for various delays during weekdays for cab botnet

Similar results are observed for the bus botnet. Figures 5.67–5.70 show the percentage of the bus botnet reachable and its responsiveness during weekdays and Sunday for various delays using the Naïve AP-selection algorithm; the other AP-selection algorithms produced similar results for the different delays. From the figures, we can see that while shorter delays somewhat decrease the botnet's reachability and responsiveness, delays of 30 seconds or more are required to significantly reduce the amount of bots able to receive commands. We also notice, from the Figs 5.67 and 5.68, that delays more than 30 seconds have minimal additional impact on the botnet's overall responsiveness. The choice of a 30-second delay defense is supported by the command propagation plots in Figs. 5.71 and 5.72 for commands injected at 4 and 8 a.m. on weekdays, respectively. While the figures only show the effect of delays on the Naïve AP-selection algorithm, the other selection algorithms produced similar results. Here, too, we find that delays of 30 seconds or more are required to significantly impair the botnet's command propagation, with further delay increases providing diminishing returns.

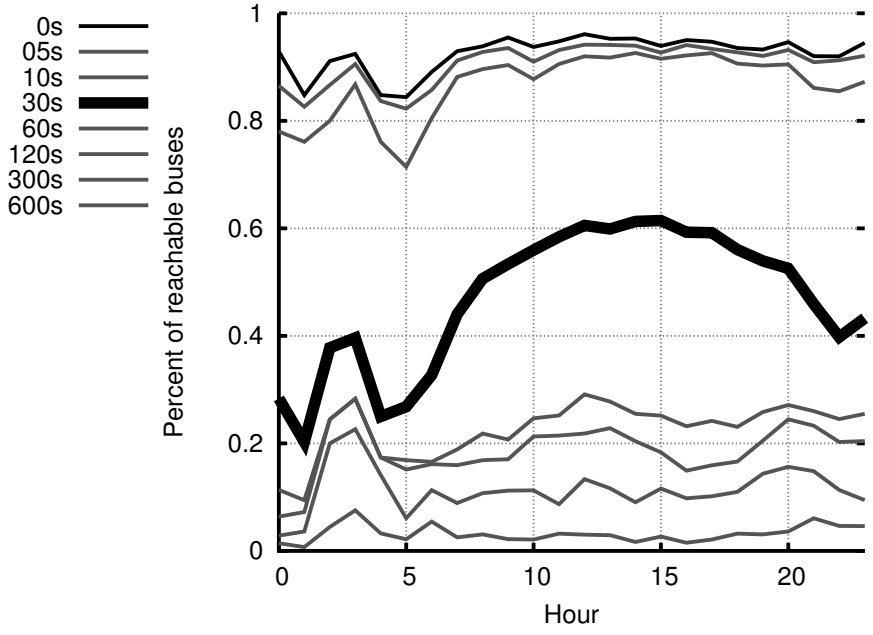


Figure 5.67: Percent of active botnet able to receive commands during weekdays using Naïve AP-selection for various delays for bus botnet

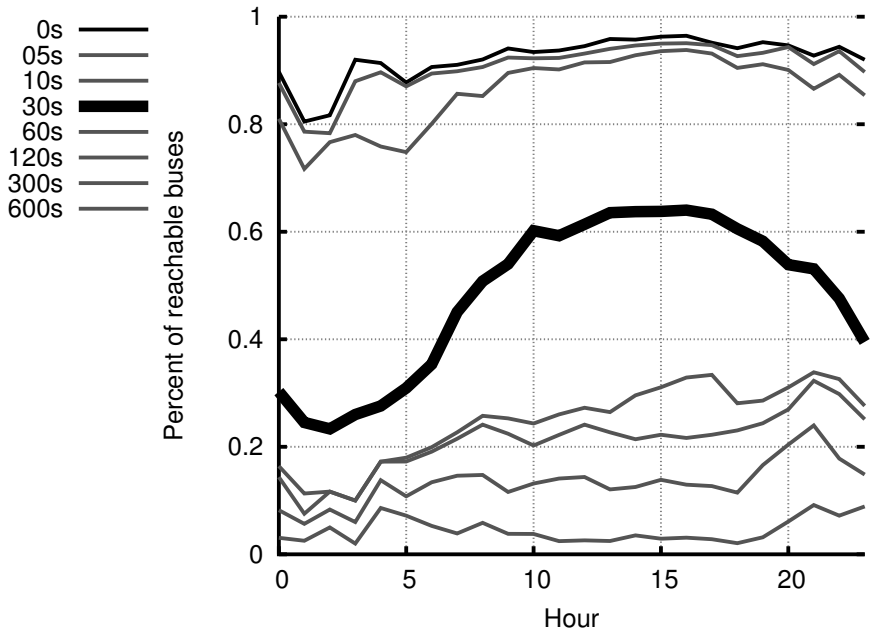


Figure 5.68: Percent of active botnet able to receive commands on Sunday using Naïve AP-selection for various delays for bus botnet

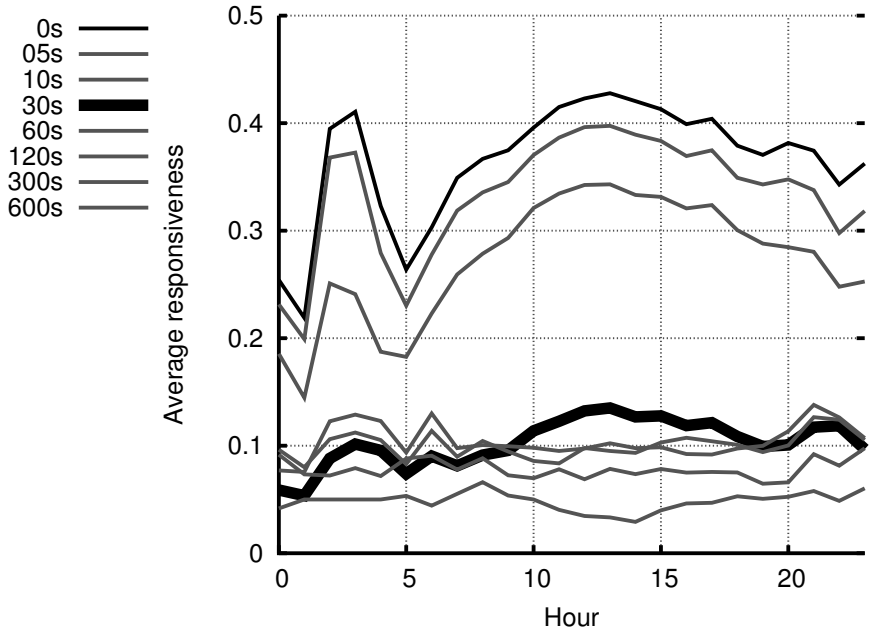


Figure 5.69: Average bot responsiveness during weekdays using Naïve AP-selection for various delays for bus botnet

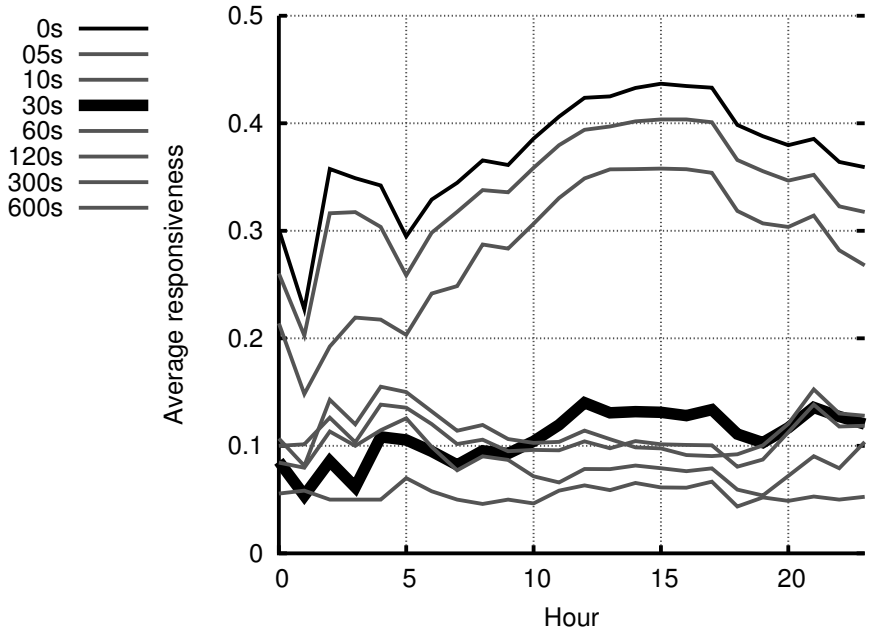


Figure 5.70: Average responsiveness on Sunday using Naïve AP-selection for various delays for bus botnet

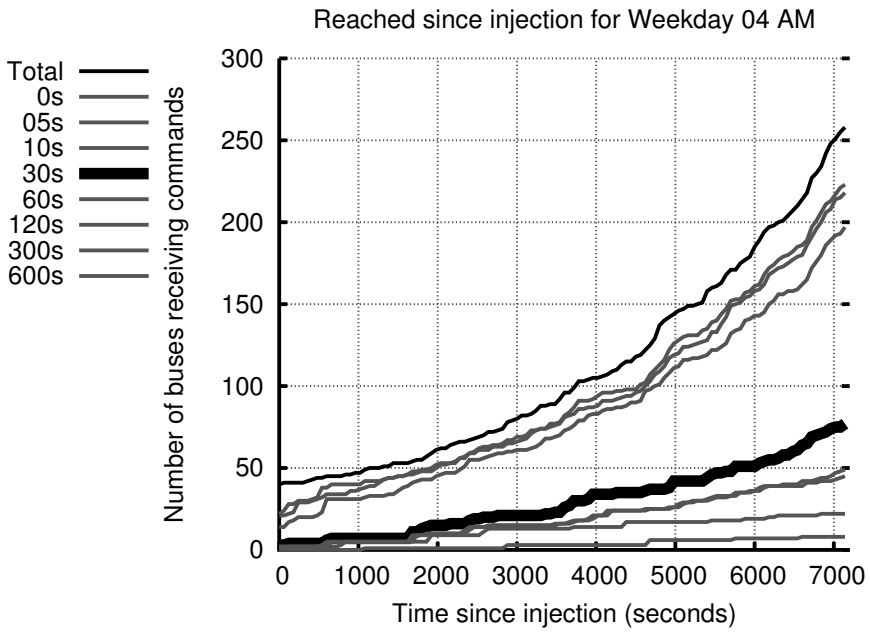


Figure 5.71: Maximum and actual propagation for a command injected at 4 a.m. during weekdays for various delays for bus botnet using Naïve AP-selection

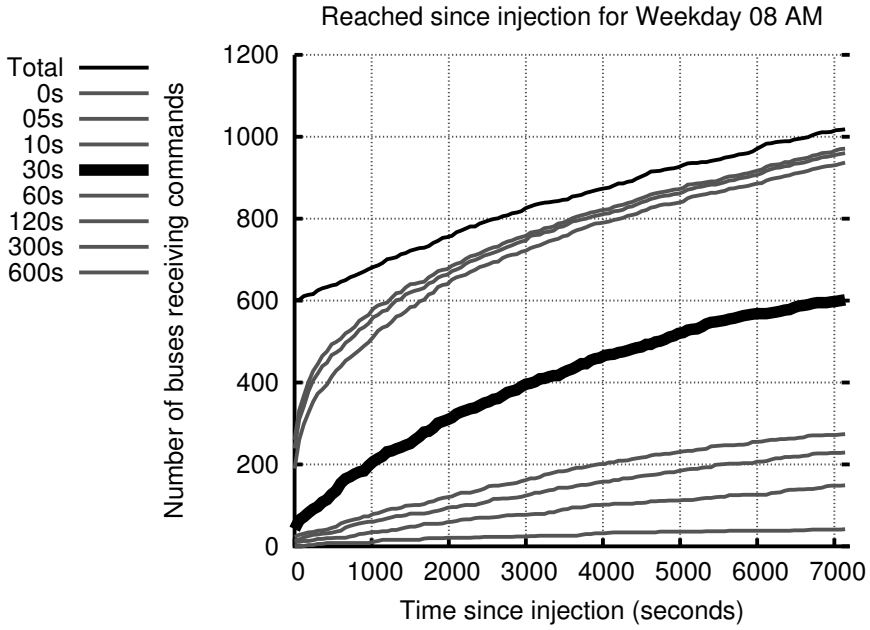


Figure 5.72: Maximum and actual propagation for a command injected at 8 a.m. during weekdays for various delays for bus botnet using Naïve AP-selection

Since considerable delays are necessary for this defensive strategy to succeed, likely annoying legitimate users, this approach is not ideal for mitigating C&C activity. However, if a delay is to be used, it appears as if a 30-second delay is the best choice for hindering the botnet with minimal impact on legitimate users.

While certainly useful, most botnets do not require a high level of responsiveness per hour, functioning effectively when able to receive commands every few hours. Meanwhile, a DDoS attack's success is directly tied to the rate it can issue SYN packets to a target victim. Therefore, we expect the delay strategy to disrupt DDoS attacks better than the C&C channel. Fig. 5.73 shows the average number of SYN packets issued by the cab botnet per hour for different delays during weekday rush hours. As expected, even small delays significantly reduce the botnet's DDoS capacity. Delays of 30 seconds and more reduce the capacity to below 300,000 SYNs per hour ($\approx$83 SYNs per second) for most hours, making the attack no longer feasible against even unprotected servers. Unfortunately, botmasters can combat this by swelling their mobile botnet ranks to increase their overall DDoS capacity. While defenders can respond by with longer delays, these will likely irritate legitimate users, limiting the applicability of this defensive measure.

For the bus botnet, we plot the effect of various delays on the amount of SYN packets for weekdays and Sunday in Figs. 5.74 and 5.75, respectively. Both figures contain plots of the various AP-selection algorithms we have examined in this chapter. As previously discovered, we find that the Naïve and Naïve patched variants perform nearly identically; likewise, the different setcover-based intelligent AP-selection algorithms produce nearly indistinguishable results. Interestingly, the improvement of the intelligent AP-selection algorithms over the Naïve variants becomes more pronounced for delays of 5 and 10 seconds, indicating that an intelligent AP-selection algorithm can somewhat mitigate the effects of a delay defense. However, we find that 30-second delays still have a significant impact on the overall number of SYN packets

sent, despite the use of intelligent AP selection, with further delays providing limited improvement. For example, between 8 a.m. and 8 p.m. on weekdays, the botnet can send between 2 and 4 million SYN packets hourly. However, with a 30-second delay defense, none of those hours can issue more than 700,000 SYN packets (≈194 per sec); that's over an 80% decrease from its peak performance of over 4 million SYN packets with no delay and easily within the capabilities of standard firewalls.

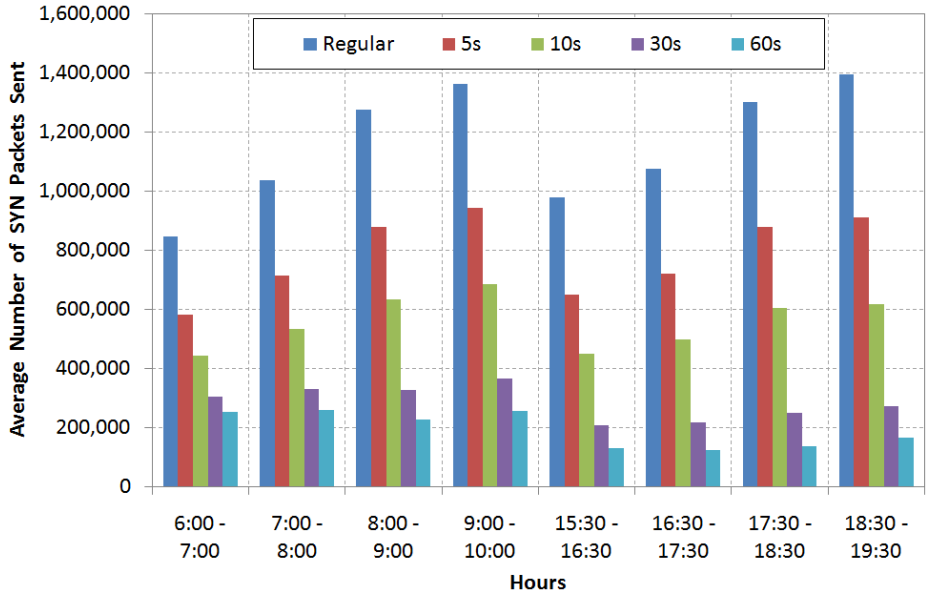


Figure 5.73: Average number of SYN packets issued during weekday rush hours by cab botnet for various delays

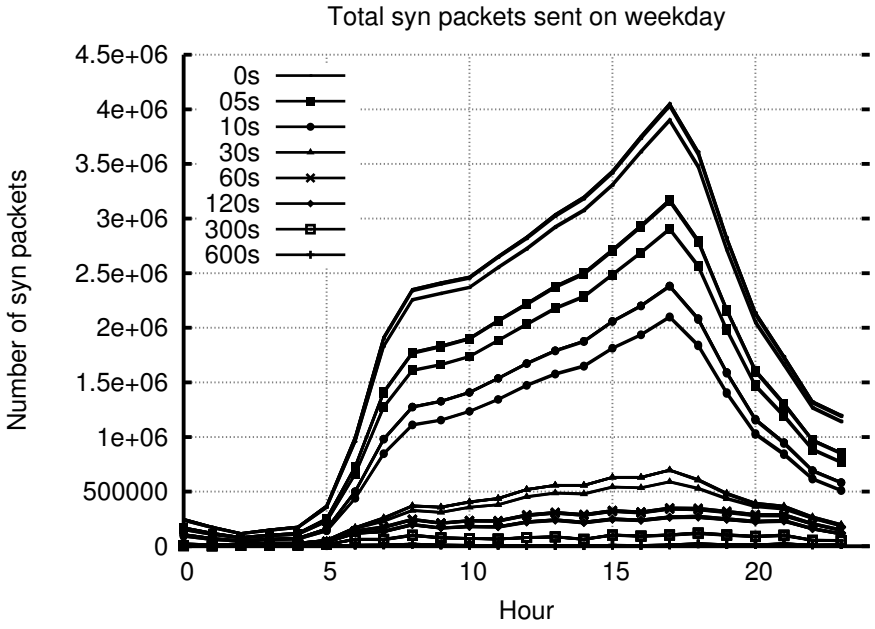


Figure 5.74: Number of SYN packets issued during weekdays by bus botnet for various AP-selection algorithms and delays

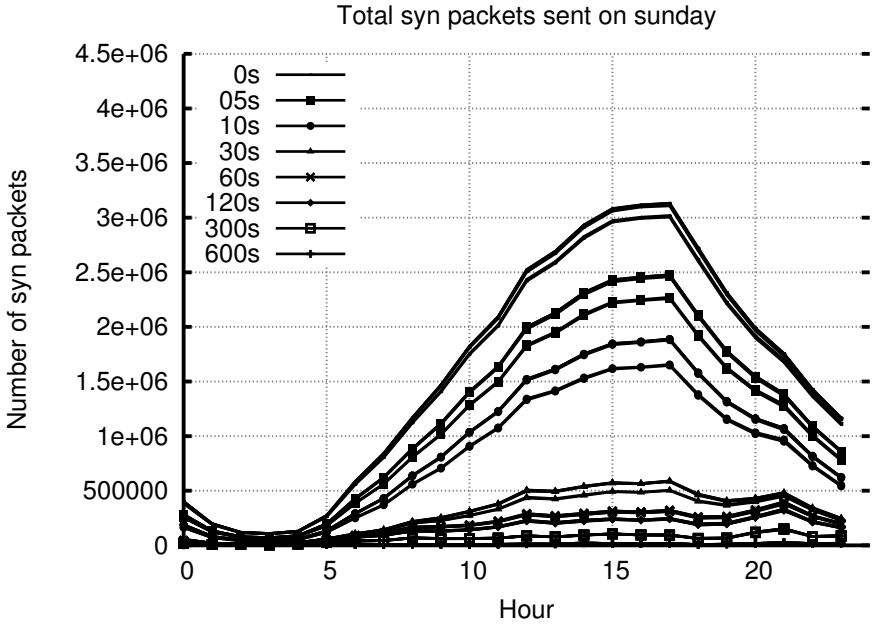


Figure 5.75: Number of SYN packets issued on Sunday by bus botnet for various AP-selection algorithms and delays

Lastly, we examine how a delay defense affects the botnet's spam attack. A spam attack's success is measured by the amount of spam the botnet is able to send. While it is unlikely that a delay defense will completely prevent the botnet from sending spam, it could significantly impact the quantity of spam sent, much like the DDoS attack's SYNs were reduced. Since spam is one of a botnet's more lucrative ventures, significantly reducing their capacity for sending spam could severely impact the botnet's revenue and spam-based infection mechanisms.

For both spam_upper and spam_lower, we plot the effect of various delays, ranging from 5 seconds to 10 minutes, on the amount of spam issued hourly by the cab botnet during weekday rush hours, Saturday, and Sunday in Figs. 5.76–5.81. As with DDoS, we find that a delay defense can successfully hinder the magnitude of spam the highly mobile botnet is able to send. For example, during the top weekday rush hours, a 5-second delay can reduce the amount of spam issued by spam_lower by over 5,000 emails and spam_upper by over 50,000. Larger delays of 10 and 30 seconds reduce spam_lower by over 10,000 and 15,000 emails, respectively, and reduce spam_upper by over 100,000 and 150,000 emails, respectively. We find similar results during the weekend. For spam_lower, a 5-second delay reduces Saturday's optimal evening hours by ≈10,000 spam emails and Sunday's optimal hour of midnight by over 5,000; for spam_upper, the amount of spam is reduced by ≈100,000 emails in both instances. Delays of 10 and 30 seconds reduce Saturday's optimal evening hours by ≈15,000 and ≈20,000 spam emails for spam_lower, respectively, and ≈150,000 and over 200,000 for spam_upper, respectively. During Sunday's optimal hour of midnight, 10- and 30-second delays reduce the amount of spam sent by spam_lower by over 10,000 and over 15,000, respectively, and reduce the amount sent by spam_upper by ≈150,000 and over 200,000 emails, respectively. While delays greater than 30 seconds continue to inhibit the amount of spam the botnet can issue, as with the other attacks, the benefits exhibit diminishing returns and are unlikely to be worth the inconvenience

they would cause legitimate users.

The effect of various delays on the aggregate amount of spam sent daily by the cab botnet are plotted in Figs. 5.82 and 5.83 for spam_lower and in Figs. 5.84 and 5.85 for spam_upper; the weekday plots are averaged across all weekdays (i.e., Monday–Friday) to keep them readable. From the figures, it is clear that 30-second delays can significantly impact the amount of spam the botnet is able to send, while delays greater than 30 seconds provide minimal improvements in countering the attack. Regardless of the day, delays of 30 seconds tend to reduce the daily amount of spam both spam_lower and spam_upper can send by ≈60–70%. For spam_lower, this equates to a reduction of ≈100,000 emails daily during weekday rush hours and ≈300,000 and ≈250,000 emails during Saturday and Sunday, respectively. For spam_upper, the results are similar but, obviously, greater in magnitude, with the daily spam during weekday rush hours reduced by ≈1.2 million emails and Saturday and Sunday reduced by ≈4 and ≈3.3 million emails, respectively. Certainly, a simple 30-second delay can considerably reduce the highly mobile botnet's spam capacity with minimal disruption to legitimate users, making it a viable and easily implemented defensive strategy that preserves the open nature of the APs.

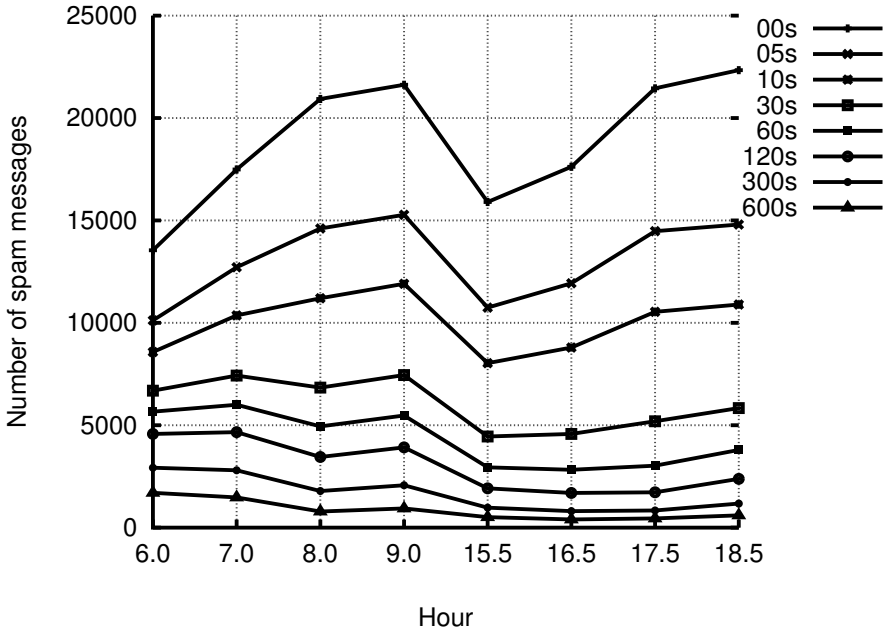


Figure 5.76: Lower-bound average number of spam emails issued by cab botnet during weekday rush hours for various delays

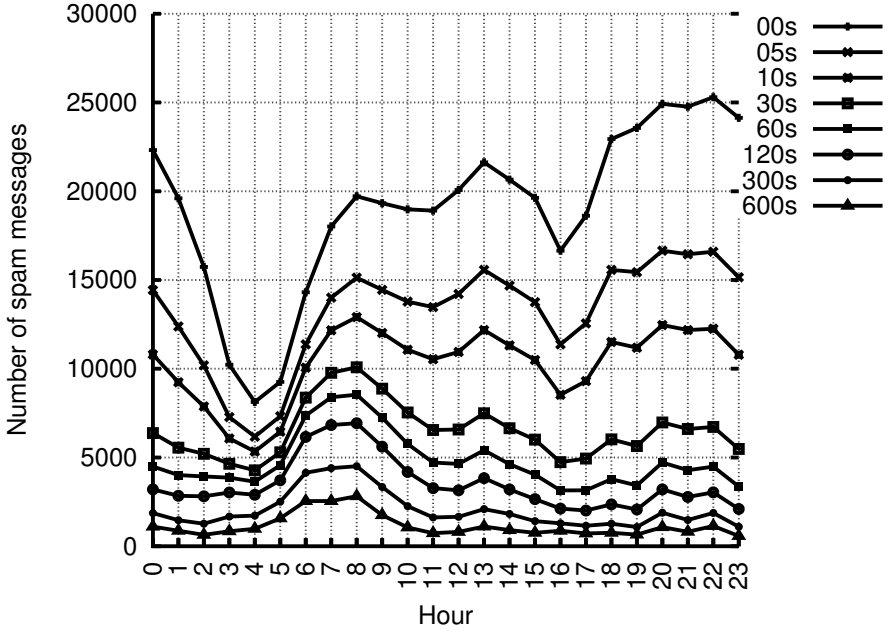


Figure 5.77: Lower-bound average number of spam emails issued on Saturday by cab botnet for various delays

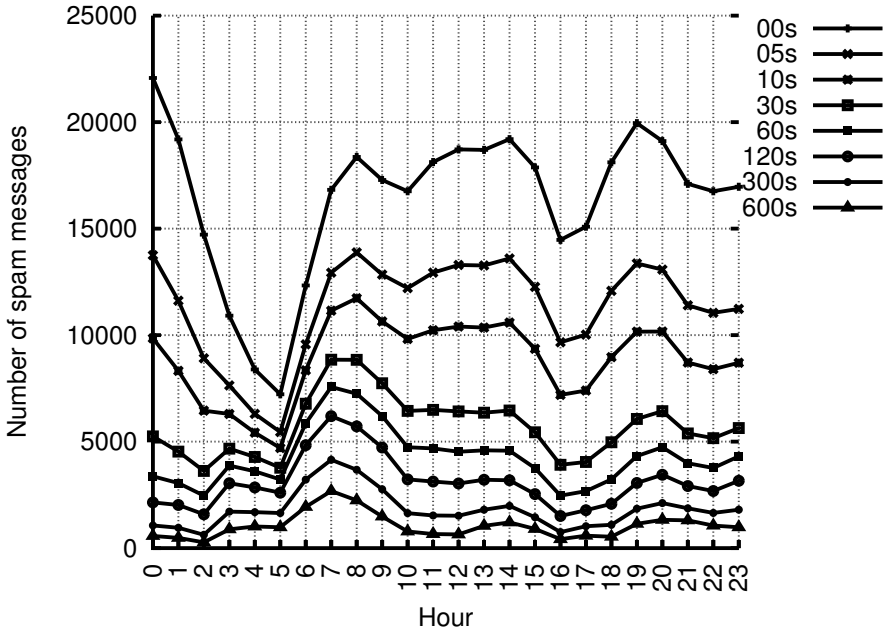


Figure 5.78: Lower-bound average number of spam emails issued by cab botnet on Sunday for various delays

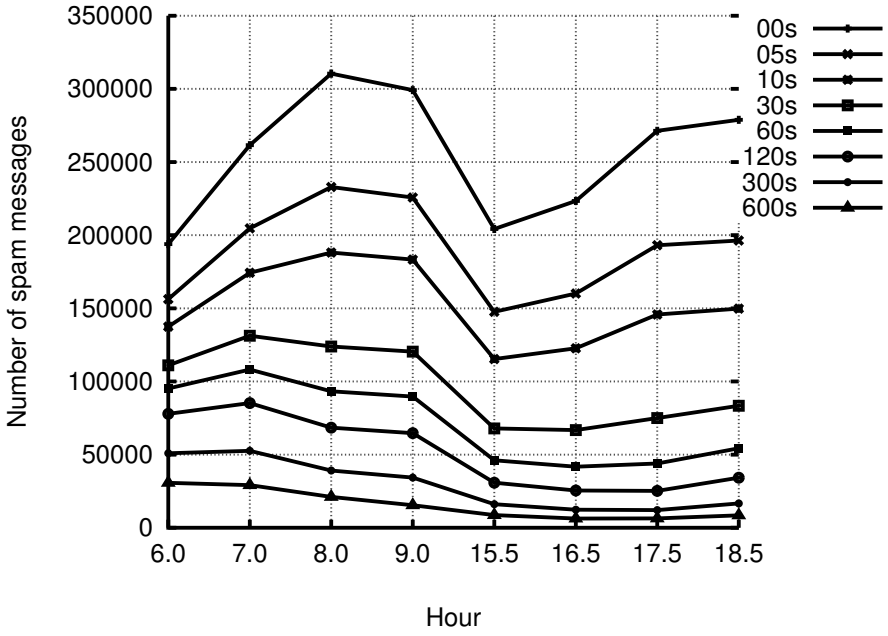


Figure 5.79: Upper-bound average number of spam emails issued by cab botnet during weekday rush hours for various delays

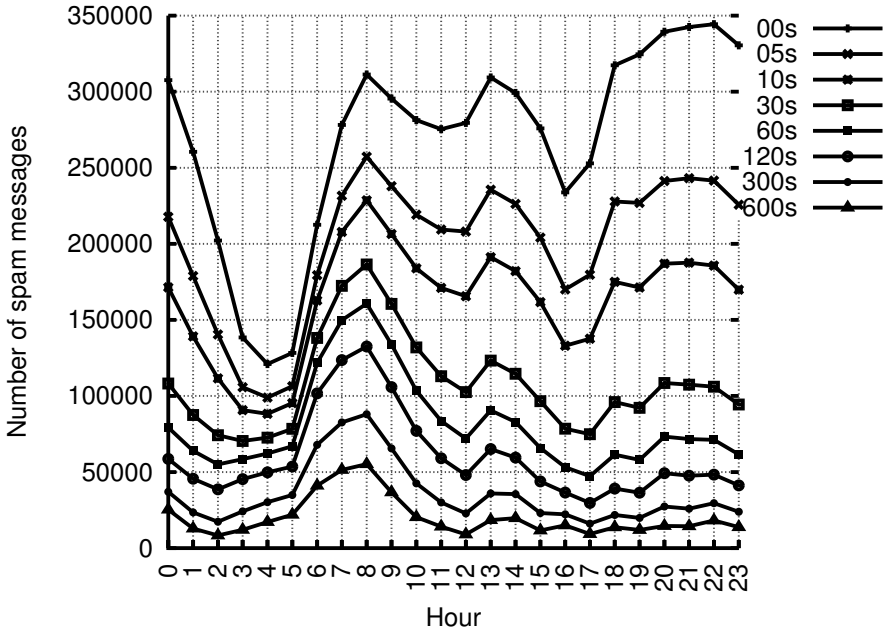


Figure 5.80: Upper-bound average number of spam emails issued by cab botnet on Saturday for various delays

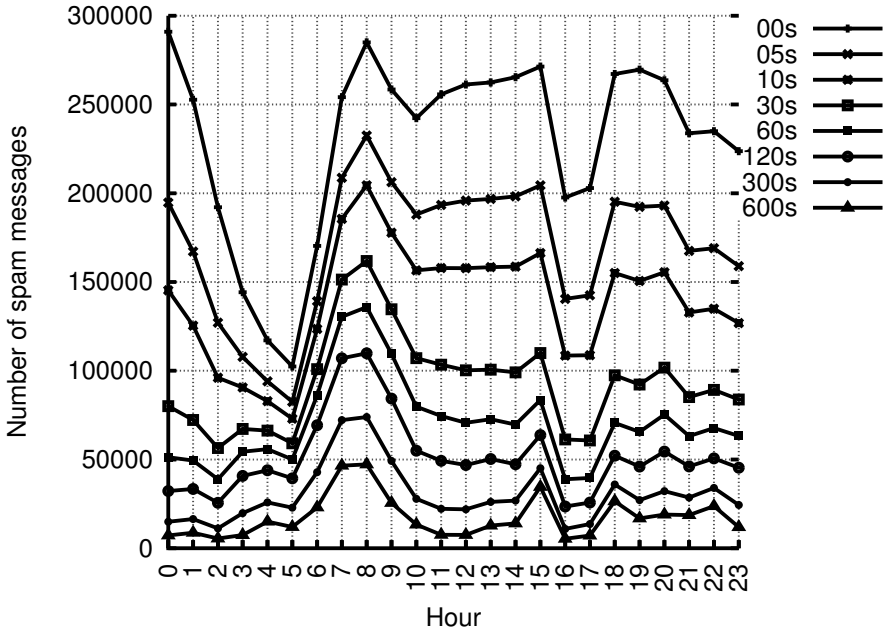


Figure 5.81: Upper-bound average number of spam emails issued by cab botnet on Sunday for various delays

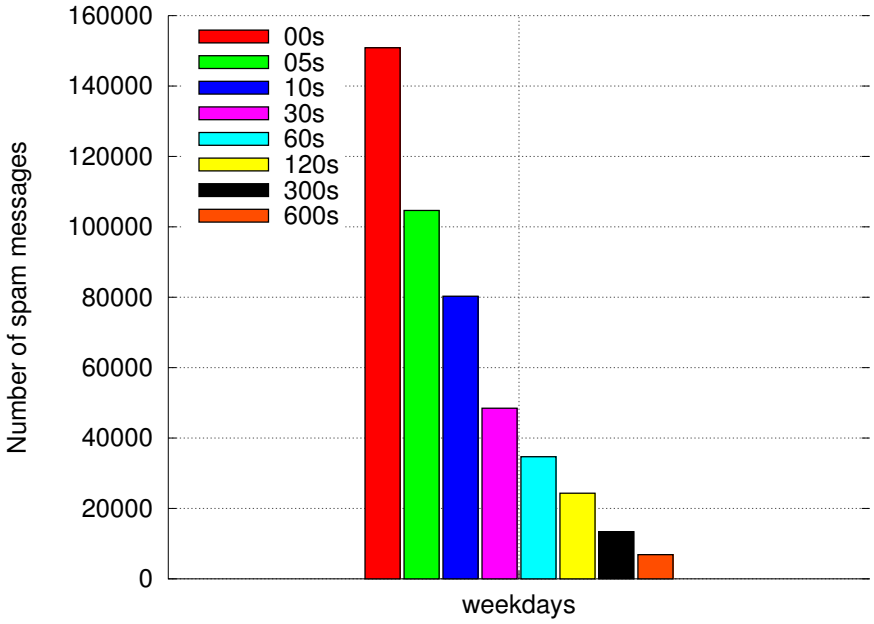


Figure 5.82: Lower-bound average number of spam emails issued by cab botnet daily during weekday rush hours for various delays

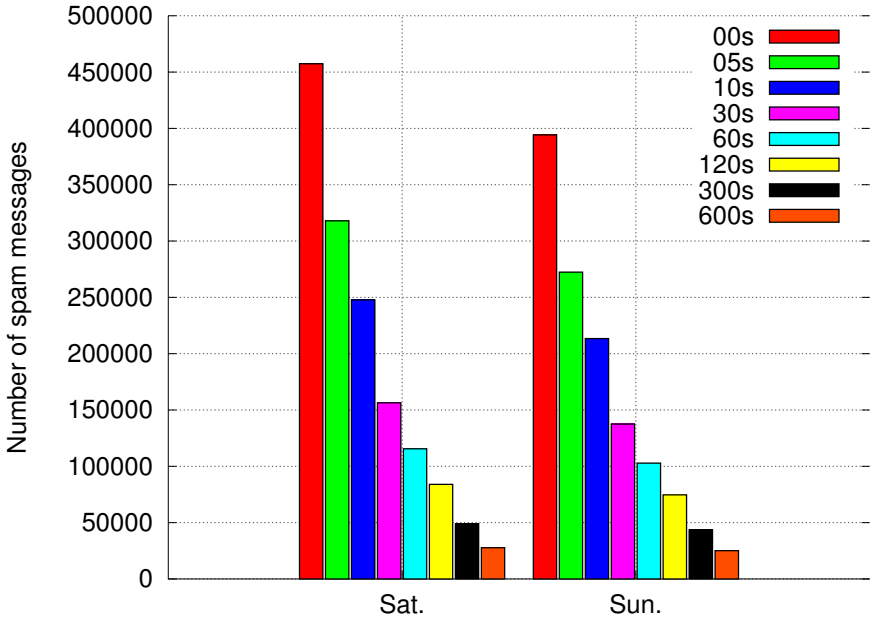


Figure 5.83: Lower-bound average number of spam emails issued by cab botnet daily during weekends for various delays

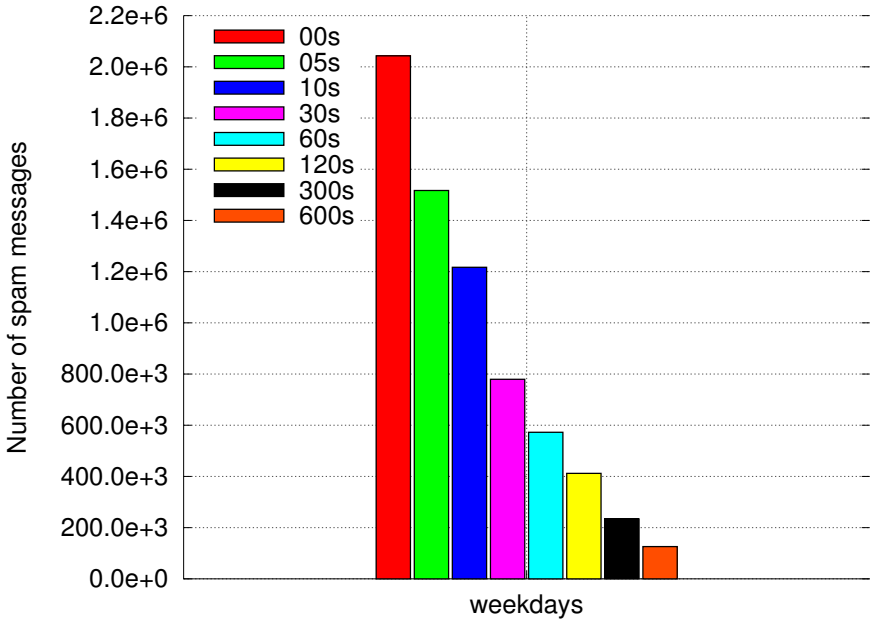


Figure 5.84: Upper-bound average number of spam emails issued daily by cab botnet during weekday rush hours for various delays

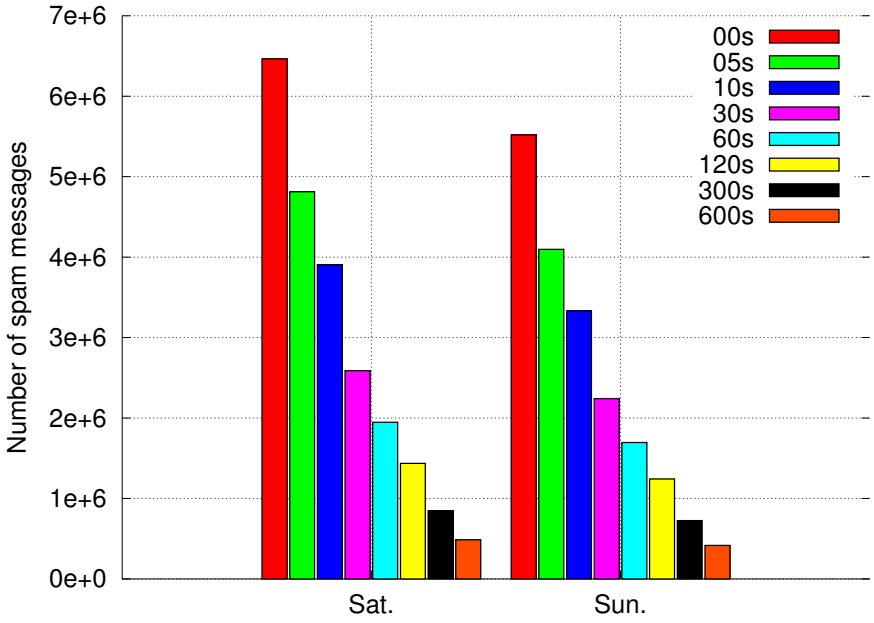


Figure 5.85: Upper-bound average number of spam emails issued by cab botnet daily during weekends for various delays

For the bus botnet, we likewise plot the effect of various delays on the botnet's spam capacity. The lower-bound amount of spam sent hourly (spam_lower) for weekdays and Sunday is shown in Figs. 5.86 and 5.87, respectively. The hourly upper-bound (spam_upper) for weekdays and Sunday is shown in Figs. 5.88 and 5.89, respectively. The daily amount of spam sent on Saturday, Sunday, and weekdays is shown in Fig. 5.90 for spam_lower and in Fig. 5.91 for spam_upper. For all the plots, both hourly and daily, we show the amount of spam sent for all the AP-selection algorithms, superimposing their results on one another to more easily discern their differences.

The first thing to notice is that a delay defense works fairly well against both spam_lower and spam_upper. Even a delay of only 5 seconds reduces spam_lower's daily spam by ≈20 thousand and spam_upper's by ≈2 million. Delays of 10 seconds more than double that reduction in spam, while 30-second delays reduce the daily spam for both spam_lower and spam_upper by ≈80%. Second, while the Naïve patched AP-selection algorithm slightly improves upon the performance of Naïve for spam_lower, it has almost no affect on spam_upper; since spam_lower it must complete the entire SMTP protocol and transfer the message data for each recipient, it best exemplifies the gains possible from the improved throughput of Naïve patched. Lastly, from the plots, we find that the more intelligent AP-selection algorithms can combat delay defenses of 5 and 10 seconds with moderate success. For example, with a 5-second delay, Weighted Setcover and its patched variant can issue ≈90% as much spam daily as Naïve with no delay, and with a 10-second delay, they can send ≈90% as much spam daily as Naïve with a 5-second delay. Interestingly, the benefits of the improved AP-selection algorithm end at delays of 30 seconds and more. While still an improvement over the Naïve approaches, it is a case of diminishing returns. From this data, we can see that even a modest delay could significantly impact the botnet's capacity to send spam. While botmasters could reduce this effect through

more intelligent AP-selection algorithms, their efforts reap little reward for delays of 30-seconds or greater.

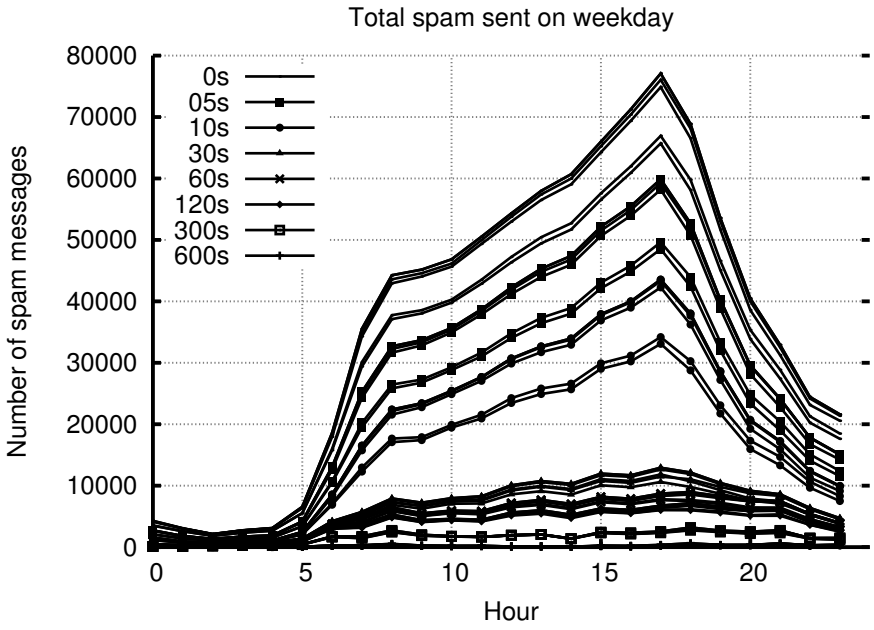


Figure 5.86: Lower-bound number of spam emails issued during weekdays by bus botnet for various AP-selection algorithms and delays

From our analysis, it appears that a delay defense could successfully disrupt a WiFi botnet. We have found that a delay of about 30 seconds seems to be ideal for disrupting the botnet while providing minimal annoyance to legitimate customers. While longer delays can further disrupt the botnet, their additional effect exhibits diminishing returns. While a delay defense is only marginally successful against a WiFi botnet's C&C channel, it holds more promise as a defensive measure against DDoS attacks and spam. However, the best defensive strategy, in terms of maximizing security while minimizing user inconvenience, might be the use of a CAPTCHA,[13] which, while easy for humans to decipher, are currently notoriously difficult—if not impossible—for bots to solve. In this scenario, once users connect to the WiFi AP, their browsers are presented with a CAPTCHA, which they must pass before being granted network access. Thus, users are not troubled with having to obtain passwords

[13]Completely Automated Public Turing test to tell Computers and Humans Apart.

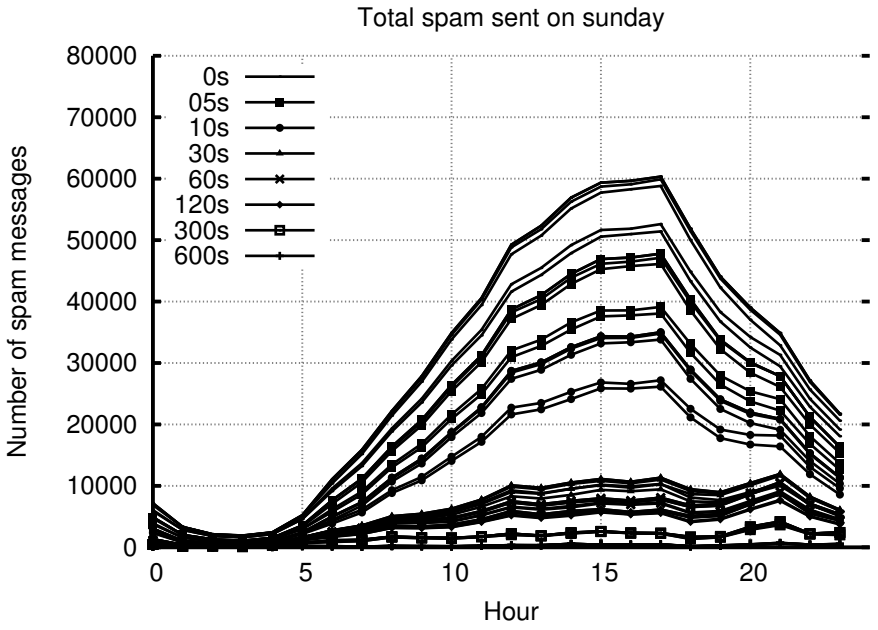


Figure 5.87: Lower-bound number of spam emails issued on Sunday by bus botnet for various AP-selection algorithms and delays

or annoyed at being subjected to potentially long delays before being allowed on the network. Because solving CAPTCHAs is difficult for computers yet easy for humans, this approach prevents bots from gaining network access while simultaneously posing minimal nuisance to legitimate users. With easily deployable CAPTCHAs available, WiFi APs requiring easy access to their networks (e.g., restaurants, cafes) should deploy this simple defensive measure before they are exploited by future mobile WiFi botnets.

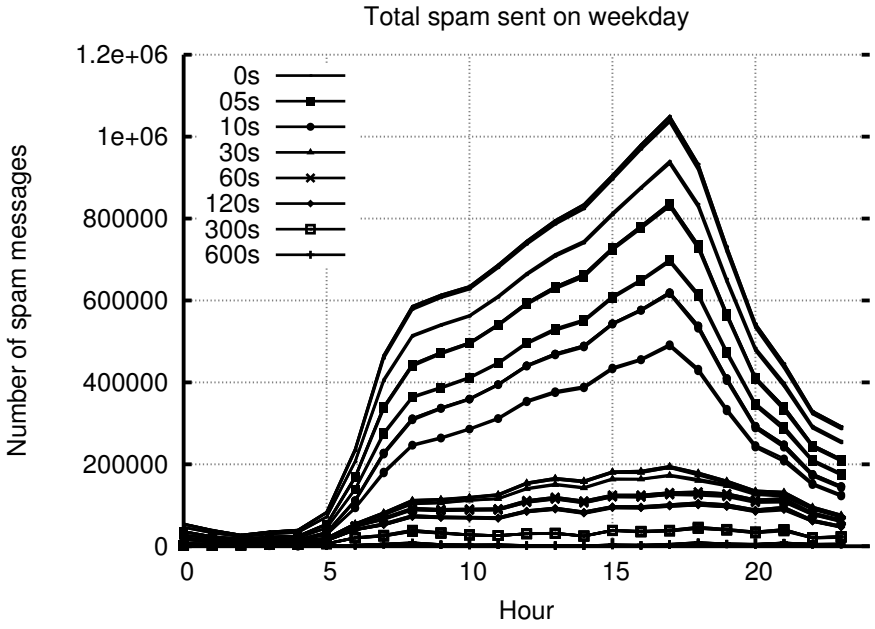


Figure 5.88: Upper-bound number of spam emails issued during weekdays by bus botnet for various AP-selection algorithms and delays

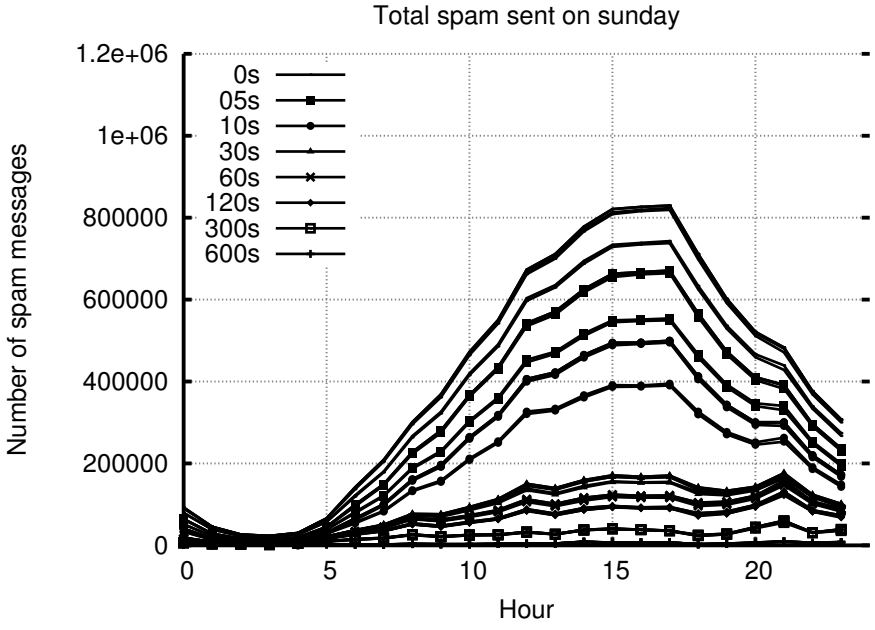


Figure 5.89: Upper-bound number of spam emails issued on Sunday by bus botnet for various AP-selection algorithms and delays

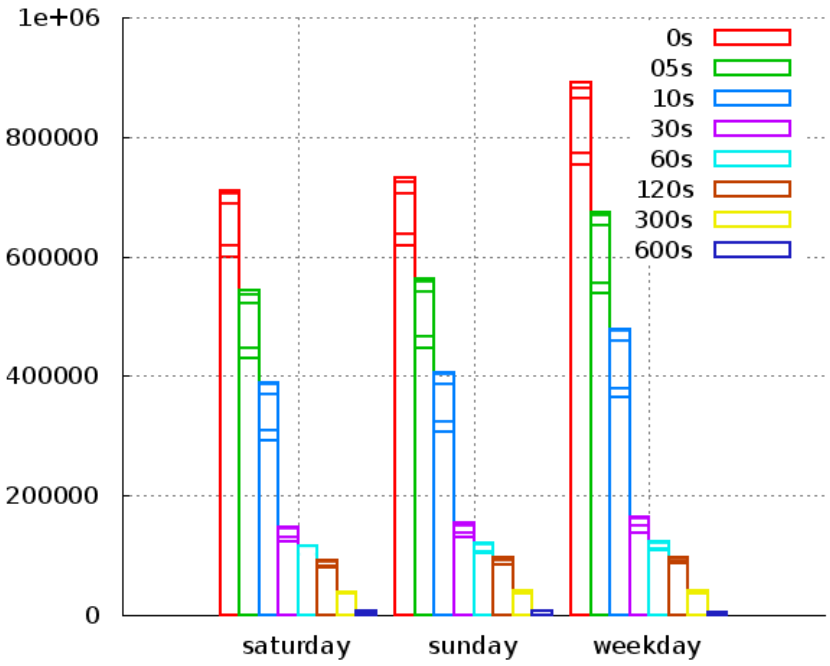


Figure 5.90: Lower-bound number of spam emails issued daily by bus botnet for various AP-selection algorithms and delays

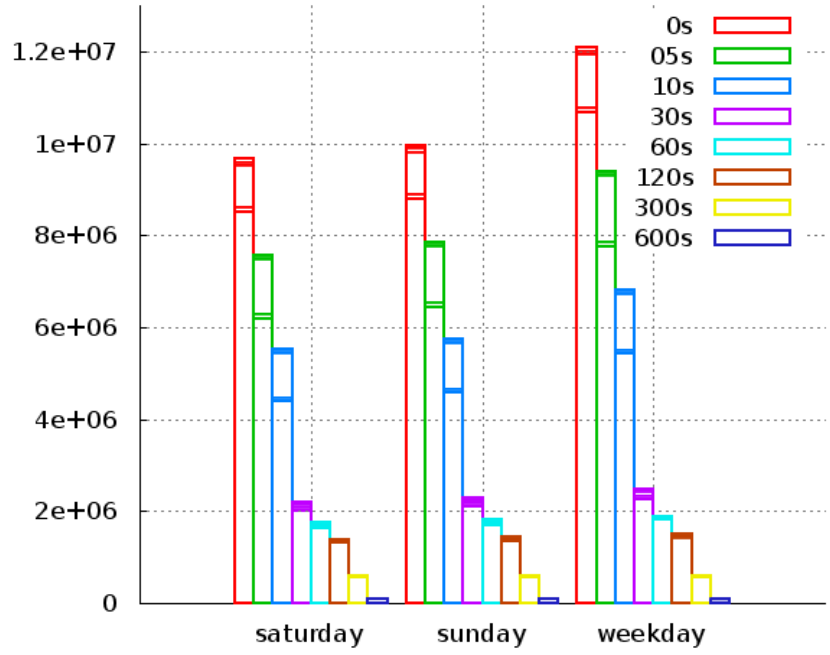


Figure 5.91: Upper-bound number of spam emails issued daily by bus botnet for various AP-selection algorithms and delays

5.7 Conclusion

In this chapter, we simulate the C&C, DDoS and spam capabilities of highly mobile botnets using only open and unencrypted WiFi networks for their nefarious actions. Our simulations make use of real-life cab mobility traces, bus routes and actual open WiFi AP locations for the urban environment of San Francisco. We discovered that a mobile botnet, whose bots are traveling quickly through an urban environment in vehicles, is capable of achieving an HTTP-based C&C channel with a fine level of control. For the cab botnet, commands typically propagate to more than 75% of the botnet within 2 hours of injection—sometimes, within as little as 30 minutes of injection. Moreover, those bots able to receive commands usually have an ≈40–50% probability of receiving them within a minute of them being issued. The bus botnet performed even better, with commands reaching over 80% of the botnet within an hour being issued, often reaching more than 90%. Like the cab botnet, bots able to receive commands have an ≈30–40% probability of receiving them within a minute of them being issued. This is a significant improvement over the Bluetooth-based C&C in [64], which achieves ≈67% propagation within 24 hours of injection. Our cab botnet has demonstrated that even a small mobile botnet of 536 devices can successfully mount DDoS attacks against unprotected systems or systems using default firewall settings. When utilizing intelligent AP-selection algorithms, our bus botnet could typically issue over 2 million SYN packets per hour (≈555 per second) and as many as 4 million (≈1,1000 per second) during its peak hour, revealing that mobile WiFi botnets possess the potential to mount powerful DDoS attacks. Furthermore, despite being limited to only open and unencrypted WiFi networks, we found that highly mobile botnets can serve as powerful spamming mechanisms. For instance, the smaller cab botnet was able to to issue ≈20,000–300,000 spam emails during peak weekday and weekend hours and between ≈400,000 and 6.4 million emails daily on weekends; even when limited to only weekday rush hours (i.e., 8 hours a day),

it could issue between ≈150,000 and 2.3 million emails daily. Similar results were observed for the bus botnet, which could send between ≈600,000 and 11.7 million spam emails daily and over 1 million per hour in certain instances. In addition, our simulations indicated that the C&C, DDoS attack and spam attack traffic were all sufficiently distributed across numerous open WiFi networks for both the cab and bus botnets, with no single open AP over utilized at any given moment. These results affirm the stealthy capabilities achievable with mobile WiFi botnets, making them especially alluring to botmasters. Finally, we examined the effectiveness of an easily implemented delay defense. While delays do little to disrupt botnet C&C, our results evince that a minor 30-second delay can significantly diminish mobile botnet DDoS and spam capabilities, with further delays providing diminishing returns. We hope our preemptive analysis of a theoretical mobile WiFi botnet has illustrated the potential threat that future mobile WiFi botnets will pose and will help foster detection and mitigation strategies specifically targeted towards this unique and powerful botnet environment.

CHAPTER VI

Conclusions

The emergence of botnets has drastically changed the malware landscape into a profitable black market. Their success, financial and otherwise, can be attributed to 4 primary properties/strategies. First, botnets must remain *stealthy* during their propagation, infection and occupation. Only through a meticulous adherence to stealth can botnets retain their numerous resources, allowing them to continue their nefarious activities and generate profits. Second, bots' *modular* nature allows botmasters to update their functionality post deployment, permitting them to alter their behavior to evade detection and promote new scams. Third, their use of a *Command and Control* channel provides botmasters the means of issuing commands to implement new scams, download binary updates or take evasive measures. Lastly, botnets require *content-delivery mechanisms*, such as botnet-based hosting service, so that new victims can reach their nefarious payload. Throughout the dissertation, we studied this stealthy aspect of botnets and its imposed limitations. In doing so, we explored botnets' primary content delivery mechanism—botnet-based hosting services utilizing FF DNS-advertisement strategies—and the future mobile botnet threatscape emerging with the increase in mobile devices and wireless connectivity.

In Chapter II, we presented the prototype RB-Seeker detection system for enterprise networks. It provides fast and automatic detection of botnet-based hosting

services utilizing redirection, employing several statistical correlation and classification techniques to analyze network traffic and DNS behavior. Two parallel subsystems (SSS and NAS) act as first line filters, cooperatively detecting redirection domains from multiple data sources. The SSS analyzes spam emails compiled from online repositories, local spam mailboxes and a spam honeypot, following embedded links to identify redirection domains. The NAS utilizes unique temporal/spatial features (e.g., inter-flow duration, flow size) of typical redirection activities to identify redirection domains without the prohibitively expensive inspection of packet payloads. These two systems feed into a third, the a-DADS, which exploits atypical DNS-query statistics of RBnet/FF domains to distinguish between malicious and legitimate domains. Evaluating RB-Seeker on real-world traces, we found it detects both Aggressive and Stealthy RBnets with low false positives. Due to the prevalent role botnets play in these RBnet/FF infrastructures, their fast and automatic detection can disrupt their hosting services and other botnet scams in which they participate. Since its data sources are readily available in most enterprise networks, RB-Seeker can be incrementally deployed and easily incorporated into existing security systems.

In Chapter III, we examined the global IP-usage patterns exhibited by different types of malicious and benign domains, including single FF (FFx1) and double FF (FFx2) domains. We deployed DIGGER, a lightweight DNS-probing engine, on 240 PlanetLab nodes spanning 4 continents, collecting DNS data for over 3.5 months on a plethora of domains. Our unique global vantage point allowed us to determine distinguishing behavioral features between domain types based on their public DNS-query results. Quantifying these features, we demonstrated their effectiveness in differentiation by building a multi-level, multi-week SVM classifier capable of discriminating between five domain types: CDN, non-CDN/MAL, FFx2, FFx1_Arec and FFx1_NArec. Applying our classifier to the larger dataset, we revealed the relative distribution of domain types in our source data and the current state of FF do-

mains, including previously unseen FFx1_NArec domains and an increased presence and range of implementation for FFx2 domains.

In Chapter IV, we extended our DIGGER detector to 320 nodes spanning 5 continents. Gathering an additional 4 months of data on new FF and CDN domains, we examined the current, state-of-art, DNS-based FF detectors, analyzing their effectiveness in detection. We developed accurate models for bot decay, online availability, DNS advertisement, and performance, which we used to evaluate novel mimicry attacks against these detection systems. Using our models, empirical evidence, and logical assumptions, we discovered that current botnet resources are sufficient to support the use of mimicry attacks against such systems. We introduced a novel spatial-detection system utilizing 5 cooperating monitoring nodes on different continents. Such a system forces botnet mimicry attacks to replicate the location-aware DNS-advertisement strategy of CDNs. Modeling the spatial-detection system, we demonstrated that, as with previous detection systems, current botnet resources can successfully evade detection at the expense of online availability and performance. We exploit this relationship by introducing a new detection metric for defending against mimicry attacks, percent connectivity, that measures the percentage of online IPs advertised. Incorporating percent connectivity into our models, we found that it can enhance existing detection systems, making them more resilient to mimicry attacks. We combined percent connectivity with our proposed spatial-detection system to produce a novel spatial-connectivity detector, discovering that even the largest botnets currently lack the resources necessary to evade detection continuously within a 24-hour period.

In Chapter V, we used real-life cab mobility traces, bus routes and actual open WiFi AP locations to simulate the C&C, DDoS attacks and spam attacks of a mobile botnet using only open and unencrypted WiFi networks; we additionally explored the effect intelligent AP selection has on botnet performance when traversing pre-

dictable routes, such as bus routes. We found that a botnet comprising mobile bots traveling quickly through an urban environment in vehicles can successfully achieve an HTTP-based C&C channel with a fine level of control. We discovered that even small mobile botnets can mount significant DDoS attacks, and when combined with traditional botnets, provide a stealthy source of additional traffic. Our results also demonstrate that mobile WiFi botnets can issue sizable amounts of spam both hourly and daily. Additionally, our simulations indicate that C&C, DDoS attack and spam attack traffic were all well distributed across open WiFi networks, with no single open AP over-utilized at any given moment. These results evince the stealthy potential of mobile WiFi botnets, increasing their appeal to botmasters. We showed that delay defenses do little to disrupt botnet C&C but can significantly impact DDoS and spam attacks, expressing diminishing returns with delays greater than 30 seconds. Lastly, we found that intelligent AP-selection algorithms did little to improve botnet C&C; however, their incorporation can improve DDoS and spam attacks and can somewhat counteract smaller delay defenses.

Throughout the dissertation, we have observed the benefits properly managed distributed systems of compromised devices can afford botmasters. Despite comprising an eclectic mix of devices with varying capabilities and online availability, botnets can intelligently leverage their available resources to achieve unprecedented proficiency in furthering their nefarious actions. Since this success is contingent on retaining control of these numerous resources, executing botnet activities in a stealthy manner has become paramount to their success and as important as the activities themselves, making stealth an intrinsic and fundamental property of botnets. Furthermore, so long as botnets can remain stealthy and retain their prodigious resources, the easier it is for them to continue to do so. Thus, it is not sufficient to focus on the currently discernible symptoms of botnet activities for detection, as these are easily modified through intelligent management of their vast resources. Rather, only through a bet-

ter understanding of how botnets attain their requisite stealth can disruption and detection attempts be improved and made more resilient to botnets' countermeasures. While stealthy operations are fundamental to botnet operations and their success, they are only made possible through the concerted efforts of their numerous constituent bots, whose available resources and online connectivity vary considerably with the diurnal usage patterns of the compromised devices' owners. Through an improved representation and greater understanding of these resources' capabilities, defenders can better exploit their inherent limitations, forcing a "Sophie's choice" on botmasters between stealth and their attack capacity—a reduction in either of which can greatly hinder botnet activities. Throughout our research, we have determined that disruption efforts specifically targeting botnet resources' limitations can significantly impact their overall success and ability for stealth. In this dissertation, these techniques have included exploiting the unreliable connectivity and diurnal usage patterns of FF botnets and the high mobility of mobile bots traveling in vehicles. In both instances, these properties are fundamental to botnet operations and cannot be easily manipulated by botmasters without sacrificing their capabilities or stealth. For example, if botmasters renounce the use of FF DNS-advertisement strategies to appear more like benign domains, the unreliable connectivity of their bots will result in most advertised IPs being offline, reducing the botnets' hosting capacity and limiting the amount of victims that reach the malicious content. Likewise, if botmasters use FF techniques and attempt to mimic the location-aware DNS-advertisement strategy of CDN domains to evade detection, the diurnal availability of their bots necessitates the use of fake or offline IPs, again reducing their botnets' overall hosting capabilities and the amount of victims they can reach. Lastly, while the high mobility of bots traveling in vehicles can grant botmasters an unprecedented level of stealth by rapidly changing which open WiFi networks portions of their attacks are carried out behind, exploiting this same property through the inclusion of delay defenses can significantly

reduce botnets' overall attack capacity. Certainly, botmasters could limit this effect through the use of other communication mediums, such as 3G/4G; however, doing so will make their attacks more apparent to cellular providers, ultimately resulting in their detection and mitigation.

Thus far, botmasters have enjoyed the benefit of knowing defenders' strategies and limitations, modifying their malicious activities to exploit weaknesses in detection efforts and continue their scams unabated. Meanwhile, botnet capabilities and limitations have remained mostly concealed from defenders, who lack the insider view necessary to take advantage of their inherent constraints. As a consequence, defenders have been forced to treat botnet symptoms and not the disease, focusing on observable and differentiating behaviors resulting from botnet activities, which can be successfully modified by botmasters to avoid detection. In this dissertation, we have contributed to the collective understanding of current and future botnet capabilities and limitations, pulling back the curtain on botnets and exposing powerful and intrinsic differentiating features and detection and disruption techniques. Providing valuable insight into the inner-workings of botnets, this work has demonstrated the symbiotic relationship between botnet activities and stealth. By exploiting this relationship and the inherent features and limitations of botnets, it has provided powerful tools to help defenders combat the botnet threat. As future work, we hope to continue this approach, further probing the resources, capabilities and limitations of existing and future botnets. Further research into future mobile botnets' many communication mediums and mobility patterns for various environments (i.e., urban vs. rural, pedestrians vs. bikers, etc.) can provide valuable insight into preemptive detection, disruption and mitigation strategies. Additionally, we hope to expand on our distributed spatial-connectivity detector, producing a viable and easily deployable detection system resilient to advanced mimicry attacks.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Capture-hpc. https://projects.honeynet.org/capture-hpc/.

[2] Crawdad: A community resource for archiving wireless data at dartmouth. http://crawdad.cs.dartmouth.edu/.

[3] Eph: Email address length faq. http://www.eph.co.uk/resources/email-address-length-faq/.

[4] Free url redirection services. http://www.emailaddr esses.com/email_url.htm.

[5] Gnu wget. http://www.gnu.org/software/wget/.

[6] Kolmogorov-smirnov test. http://www.physics.csbsju .edu/stats/KS-test.html.

[7] Netflow. http://www.cisco.com/en/US/products/ps6601 /products_ios_protocol_group_home.html.

[8] Securelist: Spam in the second quarter of 2010. http://www.securelist.com/en/analysis/204792129/Spam-in-the-Second-Quarter-of-2010/.

[9] Sfmta: San francisco municipal transportation agency. http://www.sfmta.com/cms/asystem/transitdata.php.

[10] Tiger: Topologically integrated geographic encoding and referencing system. http://www.census.gov/geo/www/tiger/.

[11] Wigle: Wireless geographic logging engine. http://wigle.net/.

[12] Netgear/atheros: 108mbps/802.11g wireless router performance testing. Technical report, VeriTest, 2004.

[13] Dns-bh - malware domain blocklist. http://www.malwaredomains.com/files/domains.txt, 2009.

[14] Phishtank. http://www.phishtank.com/, 2009.

[15] Mike Afergan. Experience with some principles for building an internet-scale reliable system. In *NCA '06: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, page 3, Washington, DC, USA, 2006. IEEE Computer Society.

[16] Mike Afergan, Joel Wein, and Amy LaMeyer. Experience with some principles for building an internet-scale reliable system. In *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*, pages 1–6, Berkeley, CA, USA, 2005. USENIX Association.

[17] P. Akritidis, W.Y. Chin, V.T. Lam, S. Sidiroglou, and K.G. Anagnostakis. Proximity breeds danger: Emerging threats in metro-area wireless networks. In *Proceedings of the 16th USENIX Security Symposium*, pages 323–338, 2007.

[18] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: characterizing internet scam hosting infrastructure. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

[19] Andreas Willig. A short introduction to queueing theory. http://www.tkn.tu-berlin.de/curricula/ws0203/ue-kn/qt.pdf, 1999.

[20] Martin Arlitt, Balachander Krishnamurthy, and Jeffrey C. Mogul. Predicting short-transfer latency from tcp arcana: A trace-based validation. In *In Proceedings of Internet Measurement Conference (IMC 2005)*.

[21] Michael Bailey, Evan Cooke, Farnam Jahanian, and Jose Nazario. The internet motion sensor - a distributed blackhole monitoring system. In *NDSS*, 2005.

[22] James R. Binkley and Suresh Singh. An algorithm for anomaly-based botnet detection. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 7–7, Berkeley, CA, USA, 2006. USENIX Association.

[23] Frank Boldewint. Peacomm.c - cracking the nutshell. http://www.reconstructer.org/, September 2007.

[24] Vladimir Bychkovsky, Bret Hull, Allen K. Miu, Hari Balakrishnan, and Samuel Madden. A measurement study of vehicular internet access using in situ wi-fi networks. In *In Proceedings of 12-th ACM MobiCom*, 2006.

[25] Kumar Chellapilla and Alexey Maykov. A taxonomy of javascript redirection spam. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 81–88, New York, NY, USA, 2007. ACM.

[26] Cisco System Inc. Ironport. http://www.ironport.com/ technology/ironport_antispam.html, 2007.

[27] Cisco System Inc. P-cube. http://www.p-cube.com/solutions/index.shtml, 2007.

[28] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: understanding, detecting, and disrupting botnets. In *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[29] Team Cymru. Ip to asn lookup. http://whois.cymru.com.

[30] David Dagon, Cliff Zou, and Wenke Lee. Modeling botnet propagation using time zones. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006.

[31] Giorgio Fumera, Ignazio Pillai, and Fabio Roli. Spam filtering based on the analysis of text information embedded into images. *J. Mach. Learn. Res.*, 7:2699–2720, 2006.

[32] Jan Goebel and Thorsten Holz. Rishi: identify bot contaminated hosts by irc nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.

[33] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[34] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *SS'08: Proceedings of the 17th conference on Security symposium*, pages 139–154, 2008.

[35] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: detecting malware infection through ids-driven dialog correlation. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–16, Berkeley, CA, USA, 2007. USENIX Association.

[36] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[37] Bruce Guenter. Spam archive. http://untroubled.org/spam/.

[38] J. Härri, F. Filali, C. Bonnet, and Marco Fiore. Vanetmobisim: generating realistic mobility patterns for vanets. In *Proceedings of VANET' 06*, 2006.

[39] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detectin fast-flux service networks. In *In Proc. network and Distributed System Security (NDSS) Symposium*, 2008.

[40] Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, 2008.

[41] Honeynet Project. Know you enemy: Tracking botnets. http://www.honeynet.org/papers/bots, 2005.

[42] Joel Hruska. Cracking down on conficker: Kaspersky, opendns join forces. http://arstechnica.com/business/news/2009/02/cracking-down-on-conficker-kaspersky-opendns-join-forces.ars, Feb 2009.

[43] Xin Hu, Matthew Knysz, and Kang G. Shin. RB-Seeker: Auto-detection of Redirection Botnets. In *Proceedings of the NDSS'09*, 2009.

[44] Si-Yu Huang, Ching-Hao Mao, and Hahn-Ming Lee. Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In *Proceedings of the 5th ASIACCS*, 2010.

[45] Ralf Hund, Matthias Hamann, and Thorsten Holz. Towards next-generation botnets. In *Proceedings of the 2008 European Conference on Computer Network Defense*, 2008.

[46] Ikee.B. http://www.symantec.com/security_response /writeup.jsp?docid=2009-112217-4458-99.

[47] D. Reed J. Oikarinen. Internet relay chat (irc) protocol. IETF, Request for Comments (RFC) 1459, May 1993.

[48] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.

[49] Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[50] Christopher Kruegel, Engin Kirda, Darren Mutz, William Robertson, and Giovanni Vigna. Automating mimicry attacks using static binary analysis. In *In USENIX Security Symposium*, 2005.

[51] P. Mockapetris. Domain names - implementation and specification. *RFC 1035*, 1987.

[52] C. Mulliner and J.P. Seifert. Rise of the iBots: 0wning a telco network. In *Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software (Malware)*, Nancy, France, October 2010.

[53] J. Nazario and T. Holz. As the net churns: Fast-flux botnet observations. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, 2008.

[54] Ross Oliver. Countering syn flood denial-of-service attacks. Technical report, Tech Mavens Inc., 2001.

[55] Chetan Parampalli, R. Sekar, and Rob Johnson. A practical mimicry attack against powerful system-call monitors. In *Proceedings of the 3rd ASIACCS*, 2008.

[56] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. FluXOR: detecting and monitoring fast-flux service networks. In *Proceedings of DIMVA'08*, 2008.

[57] Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting malicious flux service networks through passive analysis of recursive dns traces. In *Proceedings of ACSAC '09*, 2009.

[58] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, 2009.

[59] Planetlab. An open platform for developing, deploying and accessing planetary-scale services. http://www.planet-lab.org/.

[60] Honeynet Project. Know you enemy: Fast-flux service networks. http://www.honeynet.org/papers/ff/fast-flux.html, 2007.

[61] P.Traynor, M.Lin, M.Ongtang, V.Rao, T.Jaeger, P.McDaniel, and T.L.Porta. On cellular botnets: Measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'09)*.

[62] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[63] ICANN Security and stability Advisory Committee (SSAC). Sac 025 ssac advisory on fast flux hosting and dns. 2008.

[64] Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating bluetooth as a medium for botnet command and control. In *Proceedings of DIMVA 2010*.

[65] Kapil Singh, Abhinav Srivastava, Jonathon Giffin, and Wenke Lee. Evaluating email's feasibility for botnet command and control. In *In Proceedings of DSN'08*, 2008.

[66] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Chris Kruegel, and Giovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the ACM CCS*, 2009.

[67] Syamentec Corp. Norton safe web, report for livefilestore.com. https://safeweb.norton.com/report/show ?name=livefilestore.com.

[68] SymbOS.Exy.A. http://www.symantec.com/security_response /writeup.jsp?docid=2009-022010-4100-99.

[69] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM CCS*, 2002.

[70] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, June 1945.

[71] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.

[72] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *In Proc. network and Distributed System Security (NDSS) Symposium*, 2006.

[73] B. Wu and B. Davison. Cloaking and redirection: A preliminary study, 2005.

[74] Cui Xiang, Fang Binxing, Yin Lihua, Liu Xiaoyi, and Zang Tianning. Andbot: towards advanced mobile botnets. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, 2011.

[75] Vinod Yegneswaran, Paul Barford, and Vern Paxon. Using honeynets for internet situational awareness. In *HOTNETS*, 2005.

[76] Cliff C. Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *in International Conference on Dependable Systems and Networks (DSN'06)*, pages 199–208, 2006.