Security When People Matter: Structuring Incentives For User Behavior

Rick Wash
School of Information
University of Michigan
rwash@umich.edu

Jeffrey K. MacKie-Mason
School of Information
University of Michigan
jmm@umich.edu

ABSTRACT

Humans are "smart components" in a system, but cannot be directly programmed to perform; rather, their autonomy must be respected as a design constraint and incentives provided to induce desired behavior. Sometimes these incentives are properly aligned, and the humans don't represent a vulnerability. But often, a misalignment of incentives causes a weakness in the system that can be exploited by clever attackers. Incentive-centered design tools help us understand these problems, and provide design principles to alleviate them. We describe incentive-centered design and some tools it provides. We provide a number of examples of security problems for which Incentive Centered Design might be helpful. We elaborate with a general screening model that offers strong design principles for a class of security problems.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques; K.4.4 [**Computers and Society**]: Electronic Commerce—*security*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

design economics security

Keywords

incentives, design, economics, security, botnets, captcha, spam

1. INTRODUCTION

People are the weakest link in security [2]. People write their passwords on sticky notes on the screen. People don't patch their home systems and become botnet zombies. People choose whether to label a patch "critical" or just "recommended." These actions generally reflect *motivated behavior* in response to the configuration of incentives confronting individuals.[1]

Incentive centered design (ICD) is a research area with the aim of designing systems that respect motivated behaviors, by providing incentives to induce human choices that improve the effectiveness of the system. ICD proceeds from rigorous mathematical modeling of strategic interactions between people (and their systems), to practical principles for system and protocol design. ICD differs from other economics of security work in its focus on providing concrete design principles.

The design of technology systems can have a great influence over the incentives that people have to use those systems. For example, in the early days of the commercial Internet there was a debate over whether the Internet should be application-blind, allowing anyone to put anything on it, or application-aware like cable TV and online services like AOL, with central authorities filtering content [13]. ICD modeling explained why the technical architecture of application-aware networks tends to limit the number of information goods offered, and biases selection toward mass market goods. This design issue is activated again in the current "net neutrality" policy debate.

When describing how a system works we include humans as smart, distributed and — crucially — autonomous components, with their own information sets and motivations. We draw primarily on microeconomics, game theory and cognitive psychology to model incentives, individual responses to them, and inter-individual strategic awareness and behavior. Because humans are non-programmable components, we often supplement mathematical and numerical model validation methods with human subject experiments.[2]

Much ICD research has focused on two problems pivotal to information security: *getting the good stuff in* and *keeping the bad stuff out*. We describe some examples in Section 2. For instance, individuals often are not directly compensated for the benefit their actions provide to others. Using a worm-throttling technology [20] on the border of our net-

ICEC'07, August 19–22, 2007, Minneapolis, Minnesota, USA.

[1]The *motivated behavior* framing is more general than it might seem at first blush. For example, security failures due to underinformed users might be investigated as a problem in a failure to provide incentives to be better informed. Not every human action can be analyzed as a rational response to incentives, but a surprising number yield usefully to this framework.

[2]For example, one of the authors recently conducted an online experiment in "getting good stuff in" to test multiple theories of motivating public contributions to an online library[4].

work benefits others, but we may have little incentive to install it because it only aids others after a worm has already penetrated my subnet. How can we design this technology so administrators are motivated to use it? This is the problem of getting the good stuff in.

Keeping the bad stuff out is similar to pollution. It arises when an individual does not bear the direct costs that her actions impose on others. When a spy places spyware on my machine, she uses CPU and bandwidth that degrade my use of the machine and she imposes other costs by appropriating my private information. The spy doesn't take these costs into account when choosing to distribute her wares. ICD focuses attention on behavioral incentives to discover who the polluters are (they don't want to announce it!) and to discourage their polluting activities.

It is not possible to usefully summarize the full range of ICD models in a short paper. Instead, as in Section 3 we provide a more detailed discussion of a principal-agent model of screening, which can be applied to many hidden action and hidden information problems. The principal/designer sets terms and conditions for interaction with agents, who have their own objectives, and whose conduct may help or harm the principal. A successful screen provides incentives for agents to reveal their differences, so the principal can keep the bad stuff out.

2. INCENTIVE PROBLEMS IN SECURITY

Incentives issues are implicated in many information security problems. We describe a few illustrative problems drawn from two interesting, but not exhaustive categories.

2.1 Getting the Good Stuff In

Home user defenses against botnets.

Botnets, or networks of hacked machines under the control of a single attacker, have been used for a number of malicious purposes [19], including distributed denial-of-service attacks, and for sending spam and phishing emails. Botnets are possible because a large portion of computer users do not provide adequate security for their machines, either because they haven't been sufficiently motivated to learn how, or to care enough.

When a typical home user's computer is compromised for use in a botnet, he actually suffers very little; he possibly notices some system instability and network unreliability, but can easily attribute it to other external causes, such as normal variations in quality of service. He has little incentive to prevent such compromises, or to fix them once discovered, particularly since fixing the problem frequently involves reinstalling the operating system at great inconvenience. These users do not directly bear the costs of the victims of these botnet attacks, such as the downtime and lost sales for eBay. This has characteristics of an ICD problem known as the *private provision of public goods*, and earlier illustrated by the administrators who aren't motivated to install worm throttles for the benefit of external networks. What can be done to motivate individuals to contribute more effort and resources to the public good?

Privacy-enhancing technologies.

Numerous technologies "enhance privacy" by providing some level of anonymity. In general, these technologies work by making it very difficult to tell which node of a large set of nodes originated a communications. To work effectively these systems require many traffic-relaying participants [8]. However, they suffer from a common incentive problem: they rely on nodes being willing to forward anonymous messages on behalf of others. Forwarding costs bandwidth and incurs risks, since anonymous communications are often anonymous for a reason. The rational choice may be to free-ride, using the system to send messages without contributing to it by forwarding others' messages, resulting in underprovision of forwarding nodes.

A second problem relates to the size of the anonymity network. If only one user is sending traffic, then it is not difficult to figure out which user that traffic belongs to. Use of the network creates a *positive externality* by increasing the number of potential senders. However, few people actively use these networks, leaving the size of the 'anonymity set' small, since they receive little benefit from using the network when anonymity is not needed. What incentives could encourage users to provide cover traffic?

Labeling vulnerabilities.

When announcing new vulnerabilities, vendors and security providers sometimes label them (e.g., "critical") and suggest that users and system administrators use the label to determine urgency and effort. This is a clear case of an ICD problem known as *hidden information* [14]. The vendor *knows* more about these vulnerabilities than the users do, but in ICD we inquire into the *credibility* of the announced labels. Reporting a critical vulnerability in software makes the software look bad; reporting many critical vulnerabilities is even worse. This bad PR is a cost borne by the vendor but not the end users. There may be offsetting benefits from honesty, but the trade-off implies that vendors sometimes will underreport vulnerability severity.

Microsoft recently included the patch for an old vulnerability in a security update without disclosing this fact [17]. Two weeks later Microsoft released an emergency alert to install the patch because a dangerous worm exploit of this bug was published, and many systems had not installed the earlier under-labeled patch [16]. An ICD approach might design a reputation service that provides vendors with sufficient incentives to provide more informative vulnerability labels.

Knowledge workers.

In any sizable organization, knowledge workers make daily decisions that affect the security of the organization's information infrastructure. Does Bob leave his passwords on a sticky note under the keyboard, or memorize it? Will Alice just email the document instead of using the access-controlled storage system?

Careful attention must be paid to incentives for knowledge workers. Conflicts of interest between owners and employees are known in the ICD literature as *principal-agent* problems [14]. In this example there is a problem of *hidden action*: it is costly or impossible for the organization to perfectly monitor security compliance by employees. Much is known about design for such problems. For example, since appropriate security behavior is difficult to verify, incentives should be applied to other observable actions (proxies) that are closely linked with appropriate security behavior, with incentive intensity positively correlated with the informativeness of the

proxies. However, when there are multiple types of hidden action in which the organization has an interest (e.g., security compliance, mental effort, attendance to work rather than personal communications, diligence, etc.) incentives must be delicately balanced or the employee will favor some activities over others. For example, if bonuses are more dependent on timely project completion than on discovered security failures, the employee may overinvest in expedience at the expense of good security hygiene.[3]

2.2 Keeping the Bad Stuff Out

Spyware.

An installer program acts on behalf of the computer owner to install desired software. However, the installer program is also acting on behalf of its author, who may have different incentives than the computer owner. The author may surreptitiously include installation of undesired software such as spyware, zombies, or keystroke loggers. Rogue installation is a hidden action problem: the actions of one party (the installer) are not easy to observe. One typical design response is to require a bond that can be seized if unwanted behavior is discovered (an escrowed warranty, in essence), or a mechanism that *screens* unwanted behavior by providing incentives that induce legitimate installers to take actions distinguishable from those who are illegitimate.

Spam.

Spam (and its siblings spim, splog, spit, etc.) exhibits a classic hidden information problem: before a message is read, the sender knows much more about its likely value to the recipient than does the recipient herself. The incentives of spammers encourage them to hide relevant information from the recipient whether or not the email is spam to try to get through the technological and human filters.

While commercial spam is not a traditional security problem, it is closely related due to the adversarial relationship between spammers and email users. Further, much spam carries security-threatening payloads: phishing and viruses are two examples. In the latter case, the email channel is just one more back door access to system resources, so spam can have more than a passing resemblance to hacking problems.

3. SCREENING

One problem of keeping bad stuff out arises when someone has a resource and wants some users to have access to that resource, but has difficulty discerning good from bad users. The users know if they are "good" or "bad" (their type), so this is a problem of hidden information. This scenario characterizes phishing and spam, with good users being normal email senders and bad users as spammers or phishers. It also covers many other situations, e.g., are all automatic software installs and updates beneficial, or do some contain spyware? Or, are users attempting to login authorized (good) or unwanted hackers (bad)?

Proof-of-work techniques from cryptography have been proposed to keep bad users from having access to various resources; for example, there is a stream of literature proposing and debating proof-of-work schemes to prevent spammers from putting unwanted email in user inboxes. To illustrate the analytical approach to incentive-centered design, we here propose a canonical model of screening to differentiate between good and bad users, and derive some of the design principles that are conditions for a successful proof-of-work screen. Subsequently we show how these design principles are at the core of the debate in the proof-of-work anti-spam debate. In our proposed research we will likewise analytically derive design constraints that guide our design of effective mechanisms to improve home computer security.

In simple terms, a proof of work is a question such that there is a known lower bound on the amount of work (e.g., CPU cycles) necessary to answer it correctly. We now sketch a model of how a proof of work could be designed to distinguish between desirable and undesirable users. Suppose a person, Principal, has a networked computer or other similar resource. Two types of users, whom we call Good and Bad, have requested access to this resource. Principal cannot directly distinguish between them before granting access. He wants to provide access only to Good. To screen (differentiate) between them, he asks both to perform a task such as providing the correct answer to the proof-of-work question. This task is an effective *screen* if it induces Bad to act differently than Good. Once Principal can tell the difference, he can restrict or refuse access to Bad. We now characterize the incentive properties that such a task must have to be effective.

More formally, let there be two user types, Good and Bad, indexed by θ^G and θ^B, with $\theta^G > \theta^B$. To gain access, Principal requires User θ to perform a task measured in units of intensity $t \in [0, \infty]$. The task could, for example, be some arbitrary but verifiable computation that requires CPU cycles (a "proof of work"); in other applications it could be a cash payment or the provision of a password. The intensity t could represent the amount of work required. If User accesses the resource, Principal receives some benefit that can be represented by a function $r(\theta^i, t)$ with $r_\theta > 0, r_t \geq 0$ (subscripts denote partial derivatives). Benefit r represents an aggregate of all value the Principal gains from access by User. For example, User might make a cash payment for access, or the task performed may be productive work valued by Principal ($r_t > 0$).[4] How much Principal benefits from User's work may depend on whether User is Good or Bad (θ^i), and Principal may accrue other benefits from User's presence, also captured in r (for example, User may be an employee who does productive work once admitted to the resource); we assume that principal's benefit is increasing in the quality of the user (θ), meaning that Principal receives more benefit from Good users than Bad users. If we wish it is straightforward to accommodate the possibility of purely harmful users, by specifying that Bads who perform no task provide Principal with negative benefit $r(\theta^B, 0) < 0$.

Crucial to the use of proof of work to distinguish between Good and Bad is that performing the task is costly to User; we denote the effort cost by $\alpha(t, \theta)$, which depends on the intensity of task required. The cost of effort may be different for Good and Bad types (in fact, for proof of work to succeed, this must be true, and the difference in cost of effort must obey an inequality constraint, characterized below). When access is granted, User receives a benefit of s

[3]The theory does not stop at these qualitative characterizations, but guides measurement and quantitative design.

[4]In some applications, the task is dissipative, ($r_t = 0$), with its only purpose being the role it plays in distinguishing between Good and Bad users.

provided by Principal.[5]

Principal's problem is to choose the resource transfer provided to Users who gain access, and the intensity of the task required to obtain access, to maximize the system payoff to Principal. We can characterize this as defining one or more contracts that specify (s, t): "if you perform task t you will be granted access worth s." To analyze the implications of this model for designing a successful proof-of-work test, we start by considering the benchmark case in which Principal knows in advance each User's type. If so, then Principal can assign different contracts to Good and Bad users (since their type is known in advance): $\{(s^B, t^B), (s^G, t^G)\}$. For example, Bad might be granted fewer resources (low s^B) or be required to provide substantial work in return (high t^B). A truly undesirable user (e.g., a malicious hacker) could be denied access altogether (benefit $s = 0$).

In practice the user type is not known in advance by Principal. Principal's contracts offer benefit s^i to anyone who performs task t^i. Contract (s^i, t^i) is targeted at type i, though other types may choose it. By offering an appropriate *choice* of contracts (determined below) Principal may be able to screen Users so they, acting in self-interest, self-select by type into different contracts, thus revealing their type.[6] Users self-select by choosing which task t^i to perform, receiving benefit s^i when they successfully perform it. Determining these contracts is our design goal. Fortunately, by the surprising and elegant Revelation Principle [15], we can, without loss of generality, restrict the set of possible contracts over which we search to those for which User finds it in her interest (*incentive-compatible*) to truthfully reveal her type.[7]

Principal chooses the menu of contracts to maximize his total benefit subject to constraints (such as a budget constraint on the screening process). If λ is the fraction of users expected to be Good (estimated, say, from past experience), Principal designs contracts to maximize his benefit by solving

$$\max_{\{(s^B,t^B),(s^G,t^G)\}} \lambda[r(\theta^G, t^G) - s^G] + (1-\lambda)[r(\theta^B, t^B) - s^B] \quad \text{(SCREEN)}$$

$$\begin{aligned}
\text{s.t. } s^B - \alpha(t^B, \theta^B) &\geq u^0 && (1)\\
s^G - \alpha(t^G, \theta^G) &\geq u^0 && (2)\\
s^G - \alpha(t^G, \theta^G) &\geq s^B - \alpha(t^B, \theta^G) && (3)\\
s^B - \alpha(t^B, \theta^B) &\geq s^G - \alpha(t^G, \theta^B). && (4)
\end{aligned}$$

(1) and (2) are *Participation Constraints* (PC): a user must receive at least u^0 from obtaining access to Principal's resource, or will choose not to participate (attempt access), for some u^0 determined by the user's other opportunities. (3) and (4) are *Incentive Compatibility* (IC) constraints: The payoff from truthfully revealing type must be greater than the payoff from dissembling to obtain the other type's treatment. The Revelation Principle allows us to impose the IC constraints to reduce the search space for an optimal solution.

We have not specified yet $\alpha(t, \theta)$, the cost of performing a task with intensity t for user type θ. $\alpha(t, \theta)$ is determined by the choice of the task, and understanding its required properties is critical. Suppose α satisfies the following conditions:

$$\begin{aligned}
\alpha_t(t, \theta) &> 0, \quad \text{for } t > 0 && (5)\\
\alpha_{tt}(t, \theta) &> 0 && (6)\\
\alpha_\theta(t, \theta) &< 0 && (7)\\
\alpha_{t\theta}(t, \theta) &< 0, \quad \text{for } t > 0. && (8)
\end{aligned}$$

We show below that these constraints produce an effective screen that will cause users to truthfully reveal their types. The first two conditions ensure the cost of performing the task is convex. Convexity means that task cost is increasing in t and at an increasing rate. This condition is sufficient though not necessary for the main results below, as it guarantees that sufficiently difficult tasks can be created. The third condition is that for any given level of screen t, the screen costs more for the Bad than Good users.[8] The final condition is a single-crossing property we use below.

We provide intuition for the constraints in the screening problem with an illustration from a familiar security solution that is closely related to proof-of-work: passwords. By seeing the conditions required for passwords to be effective as screens, we can better understand the design constraints on proof-of-work screening systems. For this illustration, we measure the intensity (t) of the password task by the number of bits that must be correctly provided. For simplicity we assume that someone who does not know the password uses a direct search or guessing attack to obtain a valid password.

Multi-bit passwords satisfy the convexity requirement (5), (6): the number of possible passwords, and thus the guessing cost, is exponential in the number of bits. Providing a valid multi-bit password costs much less for Good (condition (7)) because Good created or was told the correct password in advance and must merely retrieve it from storage,

[5]For simplicity we assume that a transfer worth s units of value to Principal is also worth s units to a user, such as would be true for a money payment. The value transfer can be a cash payment, but may also include resource usage such as CPU cycles. It is straightforward to modify the analysis so that a transfer that costs Principal s is worth some monotonic function $f(s)$ to User.

[6]Contracts may be implicit; it is not necessary that they be formal. We overload the use of i to index contract choices as well as user types because at an optimum the Principal will offer at most one contract per type, and the each contract will be designed to appeal to a specific user type.

[7]The Revelation Principle states that for any set of contracts under which a rational agent is not truthful, there exists another set of contracts under which a rational agent wants to report honestly, and the payoffs in all states of the world are the same as in the first set of contracts. The intuition is straightforward: for whatever map agents are using to transform their true type into an announced type in the mechanism that does not induce truthtelling, imagine a different mechanism which applies that map as the agents were using before to transform truthful inputs into dishonest inputs, and then assigns the outcomes they would have gotten for those dishonest inputs in the original mechanism. This means that restricting our search to contracts where users honestly reveal their type does not limit the possible outcomes.

[8]In our setup, it is easy to think of $\alpha_\theta(t, \theta) = \alpha^G(t) - \alpha^B(t)$, the difference between the cost for Good and Bad types for a given intensity t. 7 states that this difference is positive, and 8 states it is increasing as t increases. This notation was chosen to easily generalize to more than two types of Users.

whereas Bad must use costly resources to guess it, including perhaps nontrivial timeout waits after making multiple incorrect guesses. The single-crossing property (8) requires here that the incremental cost from increasing task intensity (requiring longer passwords) is higher for Bad than for Good: the password storage and retrieval cost for Good is approximately linear, but the guessing cost for Bad is approximately exponential.

It is not surprising that passwords often work as screens — they are a well-known technique. Rather, we have used them to illustrate the mapping between some familiar properties that make for good password systems, and the abstract design constraints identified by incentive-centered design theory. Constructively, we now show that these design constraints are sufficient for a *proof-of-work* system to effectively screen out Bad users; such systems are not already widely deployed (as are password systems), nor is their effectiveness for screening universally accepted. After we formalize the design results that follow from the model above, we show that the debate on proof-of-work for spam is a debate about whether it is possible to design a proof-of-work system that satisfies the incentive-centered design constraints, and how that might be done. We propose to use these principles to guide our incentive-centered design of home security mechanisms.

Define u^i to be the total net benefit (*utility*) to a User of type i, $u^i = s - \alpha(t, \theta^i)$. Solving Principal's problem (SCREEN) yields several illuminating results:

RESULT 1. *Utility for Bad is minimal:* $u^B = u^0$.

RESULT 2. *Good gets a net gain from participating:* $u^G > u^0$.

RESULT 3. *Good users receive more value from access* ($s^G > s^B$) *but perform a harder task* ($t^G > t^B$).

Results 1 and 2 are straightforward to prove; see, e.g., [9]. Result 2 has the following interpretation: Good owns valuable property — his knowledge that he is a Good type — and must be paid an *information rent* by Principal for the use of this property; the rent is $u^G - u^0 > 0$. Result 1 is a corollary: since everyone who doesn't prove they are Good is Bad, there is no reason to provide Bad with extra surplus to reveal her type.[9]

We demonstrate Result 3 with Figure 1. The curves $s - \alpha(t, \theta) = u$ are indifference loci (user utility is a constant for all (s, t) combinations on the locus), with utility increasing to the northwest (more s, less t yields higher utility). First consider the curve $s - \alpha(t, \theta^B) = u^0$ (the indifference locus for Bad); since $u^B = s^B - \alpha(t^B, \theta^B) = u^0$ from Result 1, we know that (s^B, t^B) lies somewhere on this locus. Now construct Good's locus through (s^B, t^B); by (8) there is a single crossing and the slope is less. Since $\alpha_\theta < 0$, by (7), $s^B - \alpha(t^B, \theta^G) = u^1 > u^0$ (Good receives some surplus utility from choosing contract (s^B, t^B).) By (4), (s^G, t^G) lies to the southeast of the B locus $s - \alpha(t, \theta^B) = u^0$. By (3), (s^G, t^G) lies to the northwest of $s - \alpha(t, \theta^G) = u^1$. Thus, (s^G, t^G) must lie in the shaded area, and Result 3 obtains.[10]

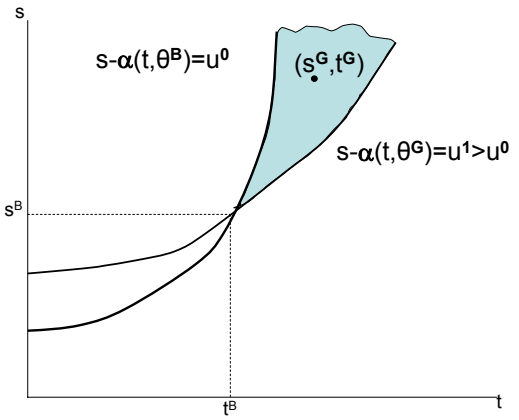

Figure 1: Screening good from bad

Assumptions (5)–(8) represent properties of the task that ensure that screening will work, and hence are principles for the design of such tasks. One of the most important design principles is not a result, but assumption (8): the incremental cost of performing the screening task must be lower for Good than Bad types (the analogous principle holds for a continuum of types). This ensures that for a given payoff, a Good type will reveal himself by a greater willingness to perform the task. If the task is equally difficult for both Good and Bad types (not satisfying (8)), then it will not differentiate users. This design principle is only sufficient to ensure such contracts exist; the specifics for a given screen still must be found.

Applications to CAPTCHAs.

We illustrate the design relevance of Results 2-3 and assumption (8) by sketching their application to another "keep the bad stuff out" problem. Challenge-response systems such as CAPTCHAs are intended to prevent automated agents from hijacking various online resources [21]. To get agents to reveal their type as human or bot, the task solving cost must be higher for the bot (presumably in CPU or programming time). Once revealed, the bot is denied access ($s^B = 0$), and its owner gets only the value from its next best alternative activity, u^0. Thus bots usually don't attempt to crack CAPTCHAs, satisfying Result 3 that the Good types exert more effort on the screening task (or, as with passwords, we could interpret t as the number of correct bits provided, which will be lower for bots if faced with an effective CAPTCHA screen).

Of course, Bad types are motivated to find ways to make the incremental solving cost similar to Good's cost (violating (8)). The development of CAPTCHA's has resulted in

[9]We have formulated the problem as if Bad can provide some positive value to Principal, albeit less than Good. If Bad types provide strictly negative utility to Principal (they are always malicious) no matter the amount of task t they perform ($r(\theta^B, t) < 0 \; \forall \; t$), then we could easily reformulate the problem with a non-participation constraint to replace (1) and a change in the Principal's objective function to correctly account for non-participation by successfully screened Bad types. The results above would be the same, and thus for discussion we assume that Bad types who are indifferent about participating ($u^B = u^0$) merely go away without attempting entry if the screen would be effective.

[10]For our password example, the task t is to provide correct bits; when a password system works, Good provides more bits than does Bad, consistent with Result 3.

improvements in computer vision algorithms[3]. To combat this, many people have come up with interesting variations that increase the cost differential between humans and computers at this task. For example, KittenAuth[11] asks humans to pick pictures of kittens out of a large set of pictures of many similar animals. Sound-based CAPTCHA's have also been used[12] as an accessible alternative to images.

Of course, the real goal of CAPTCHA's is not to keep bots out, but to keep out the malicious users who use bots to automate their interests. These people have found a creative way to provide their bots with CAPTCHA solving capabilities – outsource the CAPTCHA-solving task to humans. For example, when a bot is faced with a CAPTCHA, it might place that CAPTCHA onto the entrance page for a porn site, and the next visitor to that site solves the CAPTCHA for the bot, in exchange for (otherwise free or price-reduced) entrance to the porn site.[13] This strategy greatly reduces the relative cost to bots of solving CAPTCHAs (violating (8)), and is largely immune to technological advances in CAPTCHAs. The cost now depends on the difficulty of getting porn viewers to solve the CAPTCHA, and is out of the control of the CAPTCHA designer.

Applications to proof-of-work systems.

The ongoing literature on proof-of-work systems to block spam email are a nice example of incentive-centered design theory. In particular, although the authors do not seem to have been aware of this, the disagreements in the anti-spam proof-of-work literature have revolved around the problems of satisfying the screening design constraints.

Dwork and Naor [6] proposed a challenge-response approach to spam reduction; Gabber et al. [7] described a proof-of-work implementation that (implicitly) illustrates screening theory. Senders must perform CPU-cycle-burning tasks to obtain a valid (personal) address for a recipient. The screening task may be more costly for spammers (condition 7) because market competition requires them to run servers at full capacity, so a CPU task for every valid email address becomes prohibitive. Good agents are presumed to have sufficient idle cycles. Gabber et al. [7] magnify this important cost differential by allowing recipients to repudiate valid incoming email addresses if a single spam enters that channel, so spammers incur the CPU cost to obtain a new address for nearly every message sent, while good senders only need to pay for a good address one time.

Dwork et al. [5] and Abadi et al. [1] introduce a different class of functions that are limited by memory speed rather than CPU speed. Differences in the price/performance ratio of commercially available systems are smaller for these functions. This prevents spammers from overcoming the cost differential through strategic hardware purchases.

However, Laurie and Clayton [10] argue (again, implicitly), that this proof-of-work mechanism will fail because it does not satisfy the design constraints for successful screening. In particular, directly related to the focus of our research, they argue that the availability of botnets sufficiently lowers the cost of sending spam that a sufficiently costly proof-of-work computation would be high enough to also discourage good users from sending mail. Liu and Camp [11] responded by proposing a new design that combines a reputation system with proof-of-work. The reputation system effectively implements a form of "price" discrimination, restoring the necessary screening condition that spammers pay a higher proof-of-work cost than do desirable users.

One of us proposed a related mechanism to fight unsolicited communication using repudiable cash bonds as the screen [12]. The proof-of-work is monetary: senders put t in escrow; recipients can claim the bond or let it revert to sender. The bond cost is higher to bad types if they face a higher probability that the recipient will claim the bond; presumably recipients will not always claim bonds from good types because they want future communications from the good types. We show that such a system could make sending spam arbitrarily costly.

Extending the model.

The useful applicability of a model usually depends on the extent to which the assumptions in the model are reasonably consistent with real-world conditions. One advantage of a formal model such as we present above is the opportunity to vary the assumptions and analyze the effect on the model's predictions. We illustrate this for the application of the model to CAPTCHA tests of the sort discussed above.

Above we assumed that the Principal gets a value from attracting Good users. For example, a content web site earns advertising revenues from attracting the attention of desirable users. We also assumed that the value to Good users of gaining access had to be at least u^0 or they would go elsewhere, and that the value to Bad users of not participating was the same. This latter assumption may seem implausible for many applications. For example, if a content site faces competitors, Good users may have quite attractive alternatives, and thus have a rather high value from going somewhere else instead; let's call this value $\overline{u}$. Bad users, on the other hand, might not need to get much value from access to prefer access to this site more than non-access. For example, a spam advertiser may already have exhausted all more valuable advertising channels, and thus will be happy to get access to this site (say, to post spam in user comment entries) even if the value gained is small. Suppose the Bad users gain from access need only be $\underline{u} < \overline{u}$ for the Bad user to "participate" (wish to gain access). The question then arises: do the design guidelines for a proof-of-work system to keep out bad users change when the different user types have reservation values that differ, $\underline{u} < \overline{u}$?

Let us call the gap $\Delta \equiv \overline{u} - \underline{u}$. The answer depends on the magnitude of Δ (which is reassuring: the design results are not so fragile that they change for small differences). When Δ gets large enough, however, we start to face a problem calling for a solution known as *countervailing incentives*. This result has useful implications for applied problems.

To see how the design results change, we employ a change of variables from the definition of user payoffs: let $u^G = s^G - \alpha(t^G, \theta^G)$ be the net payoff to the Good type, and $u^B = s^B - \alpha(t^B, \theta^B)$ be the net payoff to the Bad type. Then we can rewrite the incentive constraints ((3) and (4)) in terms of the relationship between u^G and u^B:

$$u^G \geq u^B + \alpha(t^B, \theta^B) - \alpha(t^B, \theta^G) \quad (9)$$

$$u^B \geq u^G - [\alpha(t^G, \theta^G) - \alpha(t^G, \theta^G)]. \quad (10)$$

[11] `http://www.thepcspy.com/kittenauth`, Retrieved on May 8, 2007

[12] `http://www.captcha.net/captchas/sounds/`, Retrieved on May 8, 2007

[13] `http://www.boingboing.net/2004/01/27/solving_and_creating.html`, Retrieved on May 8, 2007

The new reservation utilities causes the participation constraints ((1) and (2)) to become

$$u^B \geq \underline{u} \quad (11)$$

$$u^G \geq \overline{u}. \quad (12)$$

We had from before (when $\Delta = 0$) that $u^B = \underline{u}$ (11), so it must be true from (10, 11 and 12) that

$$\Delta \leq [\alpha(t^G, \theta^B) - \alpha(t^G, \theta^B)].$$

Evaluated at the previously optimal level of proof-of-work, t^G, the right-hand side is a constant. Thus, if Δ remains small, the previous proof holds. If Δ gets large enough, the inequality (4) will not be satisfied. This happens when the net value offered to a Good type to induce her to visit the site becomes high enough. The Bad type becomes willing to do the work necessary to appear like (and get the same access as) a Good type. Technically, the Bad type's incentive compatibility constraint is not satisfied. To implement a successful proof-of-work screen when Δ is large enough, it is necessary (according to (8)) to increase the amount of work required by the Good user ($t^{G'} > t^G$). For example, if a password system is in use, the site must require harder to break passwords. If the content site is using CAPTCHAs the test must be made more burdensome.

The problem takes another twist if Δ gets even larger. We do not present the formal derivation here, but if $\Delta > \Delta^{CI}$ (where Δ^{CI} is a threshold determined by the parameters of the problem), further raising the test t^G is no longer efficient. At this point it actually becomes worthwhile to allow Bad users to get some positive value from gaining access, $u^B = s^B - \alpha(t^B, \theta^B) > \underline{u}$, contrary to the earlier result (u^B minimal). The intuition is useful for applications. When competing to attract Good users against desirable alternatives ($\overline{u}$ high), rather than make the entry screen even more burdensome, which degrades access value to Good users, it makes sense to let Bad users in (so they gain some value) but to give them a less valuable experience. For example, a CAPTCHA might be breakable by persistent spammers, but when they get in, they are permitted only low value resources (e.g., when comment spam is discovered, it is removed, so the value to the spammer is positive but small). This *countervailing incentives* result helps us understand why, even though we know it is possible to set up screens that keep out even the most aggressive spammers, most resource providers do not do so: the burden on Good users would be so high that they also would not participate.

4. DISCUSSION

Traditional information security generally deals with keeping the bad stuff out. It has been fairly successful at keeping hackers, viruses, and worms at bay. Many of the design principles that the ICD literature describes as necessary to keep bad stuff out are intuitively understood by the people who develop security technologies. Password systems are an example of technology that got this right, and as a consequence, are — if correctly used — effective at separating legitimate users from attackers. The same principles can be applied to develop new or improved solutions for both new and old problems.

Even when technical security systems are effective at keeping the bad stuff out, they may fail to get the good stuff in. Passwords again are a good example: they often fail because users are insufficiently motivated to use strong passwords that prevent password guessing attacks. Incentive-centered design can create systems that motivate users to provide desired security effort. Indeed, incentive-centered design, as an alternative to technological "hardening", may be especially effective in those applications involving agents who are not malicious, but merely undermotivated, since it is not their objective to thwart the system. Non-malicious agents may be more robustly responsive to incentives.

Incentive mechanisms to get the good stuff in are similar to those of the screening example above. However, there are some important differences that make this a novel problem. Most of the screening mechanisms listed above function by introducing artificial costs into the system (such as performing the task at intensity t). These costs are not borne equally by all — that is what makes them effective. Such costs are wasteful but necessary to induce people to reveal private information about their type. On the other hand, problems with getting the good stuff in will not be aided by additional artificial costs. People have the choice to opt out of using the system, and additional costs will just cause fewer people to use it.

Some form of positive benefit must be introduced, but in such a way that participants only receive benefits if they provide enough positive work. Once good example of such an inducement comes from the work of Luis von Ahn. For example, von Ahn and Dabbish [22] developed a game in which randomly matched users try to agree on words to describe an image. These words then are good descriptions of the image, which can then be used for search or other purposes. User contributions are incentivized through the game aspect — users experience fun from playing, thus motivating them to contribute further. Google has adopted this technology in their Google Image Labeler.[14]

Another form of benefit can be shared work. Cloudmark [18] is an email spam filtering solution that works by having users report spam messages. Cloudmark can then aggregate all of these reports and, using a reputation system, label similar messages as spam in clients' mailboxes. The work of identifying spam is being done by individual Cloudmark subscribers. Each person does a minimal amount of work, and then can benefit from the work of everyone else. This system of sharing the work provides an incentive for users to participate, as participating allows a user to also reap the benefits of the community's work. Shared work provides community value, which in turn provides the incentive to undertake the work.

5. ACKNOWLEDGMENTS

Our thinking was improved by conversations with Paul Resnick, Yan Chen, Rahul Sami, Brian Noble, Yoshi Kohno, Mark Ackerman, Marshall van Alstyne, and the rest of the ICD lab group at Michigan (Lian Jian, John Lin, Anna Osepayshvili, Kil-sang Kim, Nese Nasif, Greg Gamette, and Ben Chiao). Emilee Rader and Chris Connelly contributed greatly to readability. We are grateful for suggestions on earlier versions of this work from audiences at the USENIX HotSec06 workshop, and at the DIMACS Workshop on Information Security Economics 2007.

[14] `http://images.google.com/imagelabeler/`

6. REFERENCES

[1] M. Abadi, M. Burrows, M. Manasee, and T. Wobber. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology*, 5(2):299–327, May 2005.

[2] R. Anderson. Why cryptosystems fail. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 215–227. ACM Press, 1993.

[3] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *Conference on Email and Anti-Spam*, 2005.

[4] Y. Chen, X. Li, and J. K. MacKie-Mason. Online fund-raising mechanisms: A field experiment. *Contributions to Economic Analysis and Policy*, 5(2), 2006.

[5] C. Dwork, A. Goldberg, and M. Naor. On memory bound functions for fighting spam. In G. Goos, J. Hartmanis, and J. van Leeuwan, editors, *Advances in Cryptology – CRYPTO 2003*, number 2729 in Lecture Notes in Computer Science, pages 426–444. Springer-Verlag, 2003.

[6] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, London, UK, 1993. Springer-Verlag.

[7] E. Gabber, M. Jakobsson, Y. Matias, and A. J. Mayer. Curbing junk e-mail via secure classification. In *FC '98: Proceedings of the Second International Conference on Financial Cryptography*, pages 198–213, London, UK, 1998. Springer-Verlag.

[8] I. A. Goldberg. *A pseudonymous communications infrastructure for the internet*. PhD thesis, 2000.

[9] J.-J. Laffont and D. Martimort. *The Theory of Incentives*. Princeton University Press, 2001.

[10] B. Laurie and R. Clayton. Proof of work proves not to work. In *Workshop on the Economics of Information Security*, 2004.

[11] D. Liu and L. J. Camp. Proof of work can work. Technical report, NET Institute, October 2006. Working Paper No. 06-18.

[12] T. Loder, M. van Alstyne, and R. Wash. An economic response to unsolicited communication. *Advances in Economic Analysis and Policy*, 6(1), 2006.

[13] J. MacKie-Mason, S. Shenker, and H. Varian. Service architecture and content provision: The network provider as editor. *Telecommunications Policy*, 20(3), April 1996.

[14] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[15] R. B. Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 47(1):61–74, 1979.

[16] R. Naraine. 'detailed exploit' published for critial windows flaw. *eWeek.com*, June 26 2006.

[17] R. Naraine. Microsoft's security disclosures come under fire. *eWeek.com*, April 13 2006.

[18] V. V. Prakash and A. O'Donnell. Fighting spam with reputation systems. *ACM Queue*, 3(9):36–41, November 2005.

[19] The Honeynet Project. Know your enemy: Tracking botnets. Published on the Web.

[20] J. Twycross and M. Williamson. Implementing and testing a virus throttle. In *Proceedings of the 12th USENIX Security Symposium*, pages 285–294. USENIX, 2003.

[21] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard AI problems for security. In *Proceedings of EUROCRYPT 03*, Lecture Notes in Computer Science, 2003.

[22] L. von Ahn and L. Dabbish. Labelling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326, 2004.