Editors: Michael Lesk, lesk@acm.org
Jeffrey MacKie-Mason, jmm@umich.edu

Incentive-Centered Design for Security

Jeffrey K. MacKie-Mason
University of Michigan

Security problems are incentives problems—technology comes second. If Eve and Mallory didn't want to increase their wealth at the cost of decreasing mine, I wouldn't be at risk for identity theft. My identity doesn't get stolen and my bank account drained because Bob made a mistake in C operator precedence. Bob builds technology to keep Eve outside the gates because she wants to get in. Bob's technology might fail, but Eve's want is the driver. Why does this somewhat obvious point matter? Shouldn't we take the miscreants as a given and focus on building technology to meet the design spec, "keep out bad guys"?

In truth, motivations do matter—it's why the bad guys keep coming, and why they expend (sometimes very considerable) resources to climb over or dig under. The first fundamental design principle stems from incentives: the amount we pay Bob to build the wall higher should depend (in part) on how motivated Eve is to get inside. Human motivations matter for design in more subtle ways as well. How, for example, does Eve want to use my identity if she obtains it? Perhaps we shouldn't work so hard to stop her from getting my identity but instead design our systems so that she can't gain much once it's hers. Or maybe we should make the penalties so high for illicit use that she won't want my identity even if she can get it (the "death penalty for parking violations" solution).

Miscreants have their motivations, but so do gate builders and gatekeepers. We might pay Bob and his many relatives to build a thick gate, but Eve could easily gain entry by paying gatekeeper Carol a bribe. Users such as the castle's owner or good-hearted inhabitants like Alice have their own motivations as well. When Alice learns the password for admission through the gate, will she make the effort to keep it secure? Will she and her friends form a voluntary community to report any unfamiliar characters they see inside the gates? And if the screening system at the gate is too onerous for Alice and the other good folks, will they rise up in revolt? Further complicating this conundrum is the fact that because they're motivated, people are adaptive: whatever Bob builds, Eve will adjust her attacks, and with so many motivated (or undermotivated) people in the system, she has many potential vectors. The war goes on.

Of course, these preliminary observations aren't new. Ross Anderson argued that people are the weakest link in security.[1] The human-computer interaction (HCI) community focuses on usability and cognitive issues in system design,[2] but researchers have paid much less attention to behavioral and motivational issues (although interest is growing, see www.cl.cam.ac.uk/~rja14/shb09/). In this article, I offer the use of incentive-centered design (ICD) as a focusing lens and a principled source of constructive advice. This focus on concrete design principles distinguishes ICD from much of the recent work in security economics.

Keeping Scoundrels Out

Probably the most familiar security problem is to prevent a bad actor from getting access to a resource, whether it be information (a secret), financial, or physical. A key feature is that the scoundrel knows who he or she is before acting, but the system manager does not. This is known in economics as *hidden information* (from its origins in insurance studies, it's also called adverse selection).

Keeping scoundrels out is a common problem in social computing settings because contribution platforms tend to be open. Email, for example, is a rather open system, and self-aware spammers regularly use it to distribute unsolicited bulk advertisements. Another scoundrel problem is manipulation, such as authors who pay for favorable Amazon reviews (see Figure 1) or hackers who rig *Time* magazine's top 100 poll (see http://musicmachinery.com/2009/04/27/moot-wins-time-inc-loses/).

Familiar security technologies for keeping scoundrels out are in fact ICD solutions. Consider the use of password-controlled authentication,[3] the goal of which

What Is ICD?

ICD is an emerging, multidisciplinary research methodology for designing systems whose performance depends unavoidably on human behavior. The starting point is to recognize that human components in the system are smart, distributedóand crucial-lyóautonomous components, with their own beliefs and motivations. Humans in the loop as a performance factor have been widely recognized since the early days of aerospace engineering; only recently has a systematic effort to incorporate them in the design loop as *responsive* components made much headway.

To date, the primary contributions of motivation science to information system design have come from economics, social psychology, and game theory via theories and statistical and experimental results on individual responses to motivation and on inter-individual strategic awareness and behavior. Of course, although humans are smart devices, they arenít programmable. What we learn from motivation science are systematic, predictable (often context-dependent and always stochastic) behaviors in response to various incentives. For example, from managerial and labor economics, we know about systematic differences in the way that people respond to different types of effort compensation (piece rates, fixed salary, winner-take-all tournaments) when outcomes are uncertain. From asymmetric information economics (and evolutionary biology and psychology), we know something about how high-quality individuals can signal their desirability. From social psychology, we know predictable ways in which people respond to social comparisons and in-group identification.

We are learning that many incentive problems have systematic structure that we can exploit for design. Just as cognitive science and engineering came together in HCI, behavioral science and engineering are coming together in ICD.

is to separate good guys from scoundrels. Here, a necessary condition for success is that obtaining a valid password is too costly to be worthwhile for a scoundrel. Notice that this is an incentives proposition: it isn't impossible to obtain a password, just too costly for the scoundrel to want to incur the cost. But there's at least one other necessary incentive condition for a good password system: it must not be too costly for the good guys to obtain, maintain, and employ valid keys; otherwise, they won't bother entering the gates either! ICD solutions of this type are known as *screening mechanisms*. Proof-of-work security technologies, such as those proposed and studied to keep out spammers,[4] are also incentives-based screens.[5]

The ICD approach is illuminating when applied to a family of currently popular challenge-response system: the CAPTCHA, which is intended to prevent automated agents from hijacking various online resources.[6] A CAPTCHA's goal is to solve a hidden information problem: bots know that they're scoundrels, but hope to pass themselves off as humans. As required by screening theory, the task (recognizing graphically distorted text, for example) must be harder for bots than for humans. CAPTCHAs also display another important (and to some, surprising) feature of effective screens: the cost *incurred* is in fact higher for humans because they must solve the puzzle, whereas bots "reveal" themselves by not bothering to try.

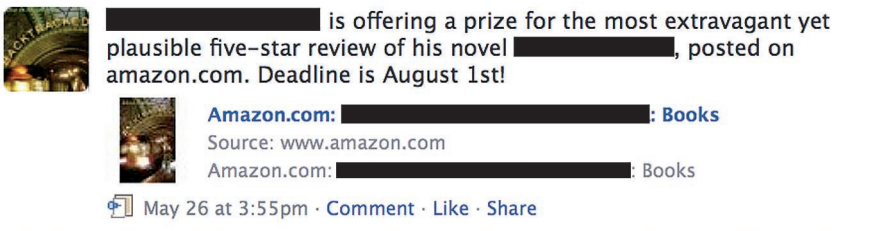

Figure 1. Manipulating Amazon book reviews. Here, the author uses Facebook to offer a prize for an extravagant review.

Another standard feature for good screens is that they satisfy the Spence-Mirrlees single-crossing condition: essentially, not only must the response cost be higher for scoundrels, but the incremental cost of responding as the challenge increases in difficulty must be higher, too. It's this latter condition that's the most crucial problem for CAPTCHA technology arms races. Consider the familiar analogue: the work cost for a good guy to provide a valid password is approximately linear in the number of bits, but for brute-force cracking, it's generally exponential. If bot creators can find ways to make the incremental solving cost for bots be of the same order as it is for good guys, game over. At this point, the only way to keep bots out is to make the challenge so tough that good guys are no longer sufficiently motivated to solve the puzzle, generating a success disaster. Much of the effort to come up with better CAPTCHA tests is aimed at increasing the incremental cost differential between bots and humans. One of the most striking attack vectors on CAPTCHAs is a direct assault on this condition: rather than improve text-recognition technology enough so that bots can solve puzzles almost as easily as humans,

some bots simply hire real humans to do the work for them.[7]

Getting Good Guys to Help

Bad guys have used botnets, or networks of hacked machines under a single attacker's control, for several malicious purposes, including distributed denial-of-service, spam, and phishing attacks. Jianwei Zhuge and colleagues[8] report that more than half of recent botnet exploits are executed through three well-known flaws for which patches were published and available for months, which only underlines a painfully familiar problem: why don't users patch their systems?

To understand the fundamental incentives problem, let's look a bit deeper. Botnets frequently hide by using a system only when it's idle. Thus, the machine's user bears little of the cost of having a zombie. He or she might not even notice the machine has been infected, and if so, there might not be much incentive to clean it. Thus, the botnet is a classic economic externality problem: others bear the costs, not the person "causing" the problem. That is, the machine owner has little incentive to make the effort to patch. Little wonder this small incentive is often outweighed by the costs of learning and implementing more secure behaviors

Many incentive mechanisms can induce people to take socially desirable action when their private incentives are insufficient. One, of course, is to fine or punish them if they don't; another is to subsidize or pay them if they do. These schemes are often worth designing into a security system, but they have shortcomings in distributed network settings: in such environments, no central authority can mete out punishments or underwrite subsidies. For these cases, another approach has received increasing attention: addressing the *private provision of public goods* problem. Generally, a public good is nonrivalrous: many can benefit from it without exhausting its value. Patched computers in a network are an example: everyone benefits from the reduced number of bot-controlled zombie machines. Incentive designers have proposed and tested various mechanisms to induce users to contribute their effort to the common good. My student Rick Wash and I, for example, are designing nonmonetary exclusion mechanisms to induce home computer users to share their security knowledge, thereby creating a "social firewall."[9] Unsophisticated users often don't know whether particular applications should be allowed or denied network access; the accumulated experience of "users like me" is valuable guidance, but those users need motivation to be willing to participate.

Discouraging Delinquents

It's sometimes desirable to keep bad actors outside of the system in the first place, but the cost of doing so isn't always worth it. Thus, it's sometimes easier to discourage them from acting (too) badly once inside. One example is the legitimate user tempted to go bad: for example, the financial manager who considers embezzling, or gatekeeper Carol who's offered a bribe to open a back door.

Some of these are known as *hidden action* (or moral hazard) problems in economics. For them, many design strategies involve assigning some share of a risky outcome to actors whose behavior is unobservable. If enough of the cost of a bad act can be shifted to the actor, then temptations with expected payoffs smaller than the cost will be foregone. Thus, we see those in the best position to harm an organization through harmful (or even simply insufficient) effort are the most dependent on bonuses and options, so they tend to lose the most from malfeasance. Some managers are fired automatically after a big enough failure on their watch.

In other situations, the delinquency might be observable but only after the action. This is largely the same as the hidden information problem: bad actors know that they're bad, but the system manager does not. However, in addition to thinking of screening mechanisms for preventing a malfeasance before it occurs, we might also consider mechanisms that punish the malfeasance sufficiently after it's discovered to deter the crime. One such security design is found in law: observe, convict, and punish. A non-governmental mechanism is to require actors to post forfeitable "bonds" (such as persistent pseudonymns for eBay's reputation service,[10] or monetary bonds to reduce spam[11]).

Another clever approach to such problems shifts the cost of the malfeasance to a third party who's in a better position to block or discourage the bad action. For example, individual computer users, in either an enterprise or at home, could have insufficient personal incentive to prevent their machines from becoming zombies. If a system manager (say, the US Federal Communications Commission) could shift the burden to enterprises or ISPs—such as by imposing a heavy fine if a denial-of-service attack is launched from machines in their domain—these organizations could provide users with the needed motivations (such as the threat of lost wages or terminated service) to resolve the problem.

All of these mechanisms face trade-offs, of course. We could raise the punishment level enough to discourage bad acts, but only at the unavoidable cost of an increase in the cost of type II errors (wrongful convictions), which is one explanation for why we don't see the death penalty for parking violations. Nonetheless, assigning

ex ante uncertain burdens to ex post observables is one important design tool in the security kit.

Technology designers often overlook incentive solutions because they tend to focus on making bad acts impossible (thicker gates) rather than on making them undesirable (cauldrons of boiling oil maintained above thinner gates). By not understanding the problem's underlying incentive structure, even when using technologies that fundamentally rely on human responses to incentives—such as passwords and CAPTCHAs—effort might be directed at the wrong feature or otherwise applied inefficiently.

Given the title of this department, I've focused on design advice we can harvest from incentive economics. But as rich and well-studied a toolkit as economics provides for incentives problems, we can learn much from other branches of the sciences of motivated behavior as well. Social psychology is particularly rich in experimental results on individuals' responses to a variety of social signals, comparisons, identifications, and so forth.

Security is fundamentally an incentives problem, so the payoffs from applying the sciences of motivated problems to security engineering will be enormous. □

Acknowledgments

My thinking on these topics owes much to my colleagues and students in the Incentive-Centered Design Lab at the School of Information, University of Michigan, especially my coauthor on security topics, Rick Wash (see www.si.umich.edu/icd/wiki/).

References

1. R. Anderson, "Why Cryptosystems Fail," *Proc. 1st ACM Conf. Computer and Communications Security*, ACM Press, 1993, pp. 215–227.
2. S.W. Smith, "Humans in the Loop: Human-Computer Interaction and Security," *IEEE Security & Privacy*, vol. 1, no. 3, 2003, pp. 75–79.
3. R. Wash and J.K. MacKie-Mason, "Incentive-Centered Design for Information Security," *Usenix Hot Topics in Security* (HotSec 06), Usenix Assoc., 2006; www.usenix.org/events/hotsec06/tech/wash.html.
4. C. Dwork and M. Naor, "Pricing via Processing for Combating Junk Mail," *Proc. 12th Ann. Int'l Cryptology Conf. Advances in Cryptology*, Springer-Verlag, 1993, pp. 139–147.
5. R. Wash and J.K. MacKie-Mason, "Security when People Matter: Structuring Incentives for User Behavior," *Proc. 9th Int'l Conf. Electronic Commerce* (ICEC 07), ACM Press, 2007, pp. 7–14.
6. L. von Ahn et al., "Captcha: Using Hard AI Problems for Security," *Proc. EUROCRYPT 03*, Springer-Verlag, 2003, pp. 294–311.
7. C. Doctorow, "Solving and Creating CAPTCHAs with Free Porn," 27 Jan. 2004; www.boingboing.net/2004/01/27/solving-and-creating.html
8. J. Zhuge et al., "Characterizing the IRC-Based Botnet Phenomenon," tech. report TR-2007-010, the Honeynet Project, 2007; http://honeyblog.org/junkyard/reports/botnet-china-TR.pdf.
9. R. Wash and J.K. MacKie-Mason, "A Social Mechanism for Home Computer Security," *Workshop on Information System Economics* (WISE), 2008; http://deepblue.lib.umich.edu/handle/2027.42/62021.
10. E.J. Friedman and P. Resnick, "The Social Cost of Cheap Pseudonyms," *J. Economics and Management Strategy*, vol. 10, no. 2, 2001, pp. 173–199.
11. T. Loder, M.V. Alstyne, and R. Wash, "An Economic Solution to Unsolicited Communication," *Advances in Economic Analysis and Policy*, vol. 6, no. 1, 2006; www.bepress.com/bejeap/advances/vol6/iss1/art2.

Jeffrey K. MacKie-Mason *is the associate dean for academic affairs in the School of Information at the University of Michigan, the Arthur W. Burks Collegiate Professor of Information and Computer Science, and a professor of economics and public policy. His research interests include the use of economics, psychology, and computer science to design information systems, including work on user-contributed content, information security, and digital goods markets. MacKie-Mason has a PhD in economics from MIT. Contact him at jmm@umich.edu.*