Economics of Software Vulnerability Disclosure

Information security breaches frequently exploit software flaws or vulnerabilities, causing significant economic losses. Considerable debate exists about how to disclose such vulnerabilities. A coherent theoretical framework helps identify the key data elements needed to develop a sensible way of handling vulnerability disclosure.

ASHISH ARORA AND RAHUL TELANG
Carnegie Mellon University

Software flaws or vulnerabilities are frequently the underlying causes of information security incidents. A recent report documents 2,524 vulnerabilities affecting more than 2,000 distinct products discovered in 2002, an 81.5 percent increase over 2001.[1] The Computer Emergency Response Team/Coordination Center (CERT/CC) received more than 4,000 reports of vulnerabilities in 2002, associated with more than 82,000 incidents involving various cyberattacks. According to the report, attackers can exploit approximately 60 percent of documented vulnerabilities almost instantly, either because exploit code is widely available or because no exploit tool is needed. Anecdotal evidence suggests that losses from such attacks can run in the millions.

Considerable debate and disagreement exist about how to disclose vulnerabilities to the public. Our theoretical framework helps identify the key data elements we need to develop a sensible way of handling vulnerability disclosure. Furthermore, we've analyzed two data sets: vendor response to disclosure and attack data from honeypots. Our results are useful for understanding how attackers respond to disclosure.

Vulnerability disclosure

The CERT/CC, when informed of a vulnerability, typically waits for the concerned vendor or vendors to provide patches or workarounds before issuing public advisories, which provide technical information about a vulnerability and patch information that users can download to protect their systems against potential exploits. Unless the vendor response is inordinately delayed, CERT doesn't publicly disclose the vulnerability without a patch. However, many believe that full and immediate disclosure is best. Full-disclosure mailing lists such as Bugtraq started popping up in the late 1990s. Unlike CERT advisories, which provide limited technical details about a vulnerability, Bugtraq disclosures frequently contain details of the vulnerabilities, as well as links to exploit code. In this article, we focus on when information should be disclosed, rather than on how much, although obvious parallels exist between the two.

Proponents of full disclosure claim that the threat of instant disclosure increases public awareness, pressures vendors to issue high-quality patches quickly, and improves software quality over time. But many believe that disclosing vulnerabilities, especially without a good patch, is dangerous. Richard Clarke, US President George W. Bush's former special advisor for cyberspace security, criticizes full disclosure, saying that releasing information before a patch is released is irresponsible and damaging (www.blackhat.com/html/bh-usa-02/bh-usa-02speakers.html#Richard Clarke).[2–4] Scott Culp, manager of the Microsoft Security Response Center, describes full disclosure as "information anarchy." The public policy problem is important, but little extant research exists to guide policy (see the "Literature review" sidebar).

Framework and model

The outcome of any disclosure policy depends on the responses of three major sets of participants: software vendors, software users, and white- and blackhat hackers. Disclosure policies affect each participant differently, so let's use our theoretical model to analyze these impacts.[5]

Figure 1 shows a product being released and used at

Literature review

Only a few articles have analyzed economic issues related to problems in information security. William Arbaugh, John McHugh, and Bill Fithen provide a vulnerability life cycle and show that many exploits occur after patches are released.[1] Stuart Schechter argues that encouraging competition among testers to discover vulnerabilities can improve quality.[2] Karthik Kannan and Rahul Telang show that a vulnerability market operated by a private firm could be undesirable because a firm would always find it profitable to disclose vulnerability information without proper safeguards.[3] Ethan Preston and John Lofton provide an overview of the current state of law and regulation for vulnerability disclosure and forcefully argue against any restriction on the publication of such information.[4] Eric Rescorla argues that finding a security hole, let alone disclosing it, is socially wasteful because the probability that a hacker would find it is very small.[5] Lawrence Gordon, Martin Loeb. and colleagues discuss how the economic issues related to information sharing in information sharing and analysis centers (ISACs) are similar to those in trade associations.[6] They also provide a theoretical analysis of how information sharing mediates the impact of security investment on expected security costs.

References

1. W.A. Arbaugh, W.L. Fithen, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," *Computer*, vol. 33, no. 12, 2000, pp. 52–59.
2. S.E. Schechter and M.D. Smith, "How Much Security Is Enough to Stop a Thief?" *Proc. 7th Int'l Financial Cryptography Conf.*, LNCS 2742, Springer-Verlag, 2003, pp. 122–137.
3. K. Kannan and R. Telang, "An Economic Analysis of Market for Software Vulnerabilities," *Proc. 3rd Workshop Economics and Information Security*, 2004; www.dtc.umn.edu/weis2004/.
4. E. Preston and J. Lofton, "Computer Security Publications: Information Economics, Shifting Liability, and the First Amendment," *Whittier Law Rev.*, vol. 24, 2002, pp. 71–142.
5. E. Rescorla, "Is Finding Security Holes a Good Idea?" *Proc. 3rd Workshop Economics and Information Security*, 2004; www.dtc.umn.edu/weis2004/.
6. L.A. Gordon, M.P. Loeb, and W. Lucyshyn, "Sharing Information on Computer Systems: An Economic Analysis," *J. Accounting and Public Policy*, vol. 22, no. 6, 2003, pp. 461–485.

time 0 (we ignore product diffusion and assume that all users start using the product at time 0). Because disclosure policy is mostly irrelevant if a blackhat discovers a vulnerability, we focus on a case in which a whitehat discovers the vulnerability at t_0.

Disclosure policies

For simplicity, we treat the disclosure policy as binary; either all information is disclosed or none. Hence, a disclosure policy is the choice of a time T, such that during that time, only the vendor receives vulnerability information. Once time T elapses, information on the vulnerability is disclosed (for example, by CERT) to the public, irrespective of patch availability. An instant disclosure policy means $T = 0$, whereas a secrecy policy implies that $T = \infty$. In terms of Figure 1, a disclosure policy T requires that this vulnerability be disclosed at time $T + t_0$. (We want to study the socially optimal trade-offs about disclosure timing. Thus, if the vendor finds the vulnerability, it will act as if the official disclosure time were infinite. If the attacker finds the vulnerability, there's no interesting policy question. Formally, this is as if the official disclosure time were zero.)

Vendors provide a patch for this vulnerability at a calendar time $\tau + t_0$, possibly after disclosure. We measure τ, T, and s from t_0, the time at which whitehats first discover the vulnerability. Attackers learn about the vulnerability at time $s + t_0$ or at time $T = t_0$, whichever is earlier. For now, unless a patch is available, we assume that attackers can exploit the vulnerability without any further delay. The expected user loss is $L_e(\tau, T: X)$, a function of T and

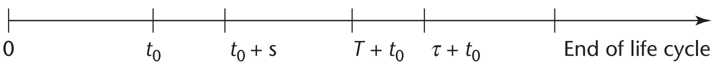

Figure 1. The software life cycle. An attacker finds the vulnerability and $t_0 + s$, and the patch is released at $\tau + t_0$.

the time window for patching, τ. Other factors that might affect expected loss are captured in X. Users suffer loss when blackhats discover the vulnerability on their own or through disclosure before the patch is available.

We define $L(t)$ as the actual cumulative user loss if users are exposed for a duration t. Intuitively, $L(t)$ should increase with exposure time t because the number of blackhats who learn about the vulnerability and the chances of an exploit tool being developed will increase. It follows that expected loss $L_e(\tau, T)$ will critically depend on when the patch becomes available (τ) and when the vulnerability is disclosed (T). Consider the following two cases: in C1, the patch is released before T; in C2, the patch is released after T.

In C1, users suffer loss only if attackers find the vulnerability before its release. In Figure 1, the attacker finds the vulnerability at $s + t_0$, and the patch is released at $\tau + t_0$. Users are attacked between $s + t_0$ and $\tau + t_0$. Hence, user loss is $L(\tau - s)$. On the other hand, in C2, attackers can find the vulnerability and exploit it for $\tau - s$. Alternatively, if attackers learn about the vulnerability when it's disclosed at T, they can exploit it until the patch is made available for $\tau - T$.

To capture the uncertainty about when an attacker will discover a vulnerability, we assume that the time that an attacker finds the vulnerability (s), conditional on a whitehat discovering it at t_0, is stochastic, with a distribution $F(s: t_0)$. Thus, the probability that an attacker won't find it within period T is $1 - F(T{:}t_0)$. We assume that $F(s : t_0)$ increases with t_0 because attackers are more likely to find the vulnerability as they accumulate experience and knowledge about the software.

Thus, we can express the expected user loss as

$$L_e\left(\tau, T; X\right) = \begin{cases} \int_0^{\tau} L\left(\tau - s\right) dF\left(s : t_0\right), \text{when } \tau \le T \\ \int_0^{T} L\left(\tau - s\right) dF\left(s : t_0\right) \\ \quad + \left(1 - F\left(T : t_0\right)\right) L\left(\tau - T\right), \text{when } \tau > T \end{cases} \tag{1}$$

The expected user loss function can take two possible forms. The first occurs when the patch is released before T but after an attacker finds the vulnerability at a time s ($s < \tau$) and exploits it for $\tau - s$. The second form applies when a patch is released after T, and an attacker finds it either before T and attacks for $\tau - s$ or at T when it is disclosed and attacks for $\tau - T$.

Given a disclosure policy T, the vendor decides how to allocate its resources to develop and test the patch. The vendor's objective function (modeled here as a cost function to be minimized) has two terms. The first is the cost of developing the patch, $C(\tau)$. All else held constant, the quicker the patch, the higher the development costs—that is, C is a decreasing function of τ.

The second cost, λ, is a proportion of user loss that the vendor internalizes (through reputation loss or a loss of future sales). Currently, vendors aren't legally liable for user losses arising from product vulnerabilities, but this could change. When $\lambda = 1$, the vendor internalizes the entire loss to users, so the interests of the vendor and society at large are perfectly aligned:

$$V = C(\tau) + \lambda L_e(\tau, T). \tag{2}$$

The social cost is simply the sum of patch-developing cost and loss to users:

$$S = C(\tau) + L_e(\tau, T). \tag{3}$$

Model limitations and possible extensions

For simplicity, we assume the vendor makes a one-time, committed decision about when to issue a patch. Additionally, patching time for our purposes is deterministic, and we assume the patch's quality is fixed. Eric Rescorla has shown that users don't apply patches immediately after they're available,[6] so we can extend our framework[5] to a case in which not all users install a patch immediately on its release—only a fraction $p(x)$ of users apply the patch at time x where $x > \tau$ and $p'(x) > 0$ and where the probability of patching depends on the patch's quality, which the vendor decides. Other extensions include explicitly allowing users to protect themselves without a patch or letting vendors make real-time decisions (from time to time) about when to develop or release an already-developed patch in response to environmental changes (for example, news that blackhats have exploited the vulnerability).

However, this model ignores two important aspects of the disclosure debate. Full disclosure proponents argue that disclosure pressures vendors to ultimately improve their software's quality. Our model deals more with the ex-post release of patches than ex-ante software quality. Additionally, disclosure might educate programmers or network administrators on new classes of bugs and improve the overall security level.

However, the model captures important elements of the trade-off, such as the cost of rushed patching to vendors $C(\tau)$. We can use the model to analyze how disclosure policies affect vendor behavior, and the factors that condition their responses, such as patching costs, likelihood and severity of user loss, vulnerability characteristics and legal liability (if it were to be in place), and vendor characteristics (for example, open source versus closed source). Most important, given the behavior of blackhats and users, as captured by $F(s)$ and $p(x)$, we can ask two questions: What is the value of T that maximizes social welfare (minimizes social loss), and how can a social planner use T to force the vendor into issuing patches in an optimal fashion?

Key implications

One implication of our framework is that vendors respond to quicker disclosure with quicker patches. Thus, instant disclosure would force vendors to issue patches faster, although this policy is rarely optimal.

If users don't patch their systems quickly, then the optimal policy is to give vendors more time. In an extreme case, if a large enough fraction of users never patch, then a secrecy policy is optimal.

If vendors decide to issue a patch on a real-time basis (rather than making a commitment to patch earlier), the optimal policy is to not disclose the vulnerability or issue a patch until a hacker finds it. After this, the vendor can issue a patch immediately.

For a new or recently released product, a larger disclosure window is useful; because the product isn't extensively understood, the exploit tools aren't widely available. Moreover, if $F(s)$ (the probability of hackers finding it conditional on a whitehat having found it) were very small, as Rescorla argues,[7] then the optimal policy is to never disclose the vulnerability.

As the internalization factor (or, someday, legal liability) increases, vendors are more aggressive in patching their systems. Similarly, the optimal disclosure window also shrinks.

Empirical studies

We conducted two empirical studies using our model. In the first, we investigated the time it takes for vendors to patch a vulnerability in their systems and tested whether this time systematically differs if the vulnerability is disclosed versus if it's kept secret. In the second study, we examined the patterns over time of attacks experienced by a host when the vulnerability is secret, disclosed, or patched.

Vendor response to the disclosure policy

Our framework helps us understand how vendors might behave under different disclosure policy regimes. In particular, we found that instant disclosure without a patch might increase customer loss. If vendors had to internalize some of the costs, they might have incentive to provide patches early. The exact timing of the patch release also depends on the patching cost. If this cost were trivial, we might not see significant differences between the two policies for some vulnerabilities. Although we didn't observe the cost of patching, we did observe the vulnerability characteristics and vendor type (public firm, open-source vendor, vendor size, and so on).

At the Heinz School at Carnegie Mellon University, we collected the data from CERT/CC and the Bugtraq mailing list.[8] As noted earlier, unless the vendor response is inordinately delayed, CERT doesn't publicly disclose the vulnerability without a patch. On the other hand, many vulnerabilities are posted without a patch (and sometimes with the exploit code) on Bugtraq. As a first approximation, Bugtraq corresponds to instant disclosure, whereas CERT typically gives the vendor at least 45 days to provide a patch.

We collected 504 observations from both sources and examined how the decision to patch—and when—changes with the policy, controlling for various vulnerabilities and vendor characteristics. The data presents methodological challenges. For example, in many cases, whether a given vendor's products are affected by a certain vulnerability is listed as "unknown" by CERT. Further, the number of vendors CERT lists to be actually or potentially vulnerable is often larger than the corresponding number in a Bugtraq disclosure. To address this, we first used a joint sample from CERT and Bugtraq to estimate the true number of vulnerable vendors in the unknown category. Then, we estimated a *probit* model (commonly used when the dependent variable is binary) to examine how the probability of vulnerabilities being patched differs between two policy regimes. In addition, we used the proportional hazard model (Cox PHM[9]) to estimate differences in the speed of patching between two regimes.

The average patching time was 242 days for CERT ($N = 186$) and 390 days for Bugtraq ($N = 318$). Approximately 60 percent of vendors whose products were affected by vulnerabilities disclosed exclusively in Bugtraq patched, whereas the corresponding number is 77 percent for vulnerabilities disclosed exclusively in CERT. When we controlled for other factors, we found that both the probability and the speed of patching were approximately the same in both regimes. This finding is surprising because the results don't support the claim that full disclosure pushes vendors into developing patches more quickly. One limit of our research was that we didn't control for patch quality. It's also plausible that vulnerabilities disclosed exclusively on Bugtraq could be less significant, in an economic sense, than those disclosed on CERT. Subject to this qualification, our results suggest that if benefits exist for early disclosure, they must lie in letting users protect themselves without a patch and perhaps in the incentives for vendors to develop more secure products in the first place. We also found that open-source vendors were quicker to patch than closed-source vendors and that more severe vulnerabilities (as per the Common Exposures and Vulnerability [CVE] ICAT database) were patched faster. More recent vulnerabilities (after the attacks of 9/11) were patched more often and more quickly.

Attacker reactions to disclosure

In the second study, we empirically explored the impact of vulnerability information disclosure and patch availability on several attacks seeking to exploit the vulnerability. We collected data mainly from 14 honeypots running in different locations and operating in different environments—Linux, Solaris, OpenBSD, and Windows—for several weeks in a year.[10] Unlike real networks, where distinguishing between an attack and legitimate traffic isn't always possible, honeynets provide an easy way to detect attacks because honeypots don't have legitimate network traffic.

We created our key variable—the number of attacks

Our results are also consistent with other findings that indicate a significant fraction of users might not install patches promptly.

targeting a vulnerability—by matching attack data with attack traffic signatures obtained from publicly available sources such as www.whitehats.com and www.snort.org/cgi-bin/done.cgi. We selected vulnerabilities randomly from the CVE ICAT database of software vulnerabilities. We classify vulnerabilities as *secret* (neither published nor patched—although all secret vulnerabilities were eventually disclosed), *published* (published but not patched), or *patched* (published and patched). A given vulnerability could go from being secret to published to patched during the course of the study period.

The data consisted of 2,952 observations during nine weeks from November 2002 to December 2003 for 328 different vulnerabilities. Of those vulnerabilities, 77 were left unpatched, 160 became public the same day a patch was released, and 76 were patched afterward. Only 44 vulnerabilities were exploited in our data. We found that, on average, secret vulnerabilities ($N = 24$) received 0.32 attacks per host per day, whereas published vulnerabilities ($N = 77$) attracted 5.45 attacks per host per day. Patched vulnerabilities ($N = 233$) attracted 2.5 attacks per host per day.

We found that publishing vulnerabilities attracted attacks, which declined with time. Patch releases also attracted attacks initially, which then declined. When averaged over time, the results imply that disclosure results in more attacks, but patching reduces them. Secret vulnerabilities suffered the least number of attacks. It's tempting to interpret this as evidence in favor of secrecy, but if a secret vulnerability suffered several attacks, it would quickly become public, so the average number of attacks for a vulnerability that remains secret over time must, by definition, be small. Moreover, the study analyzed observed attacks rather than economic losses per se; although secret vulnerabilities are naturally attacked less frequently, such attacks could result in greater damage. The finding that a patch release is also associated with a spike in attacks suggests that perhaps a patch itself provides valuable information to hackers and that attackers expect that users won't patch promptly enough.

Our empirical results suggest that the evidence supporting the case for instant disclosure, namely that it provides a strong incentive for vendors to develop patches more quickly, remains weak. Our results are also consistent with other findings that indicate a significant fraction of users might not install patches promptly. Furthermore, disclosure on Bugtraq sometimes occurs without a patch—and in some cases, exploit code might even be provided—so our results don't support full and instant disclosure as the optimal policy. Put differently, our results imply that full and instant disclosure can be socially optimal only if it either prompts vendors to develop more secure products or lets users protect themselves in other ways, including workarounds or additional perimeter defenses. Both questions require additional empirical research.

Our discussion raises the fundamental question of who "owns" vulnerability information. Some authors have argued for creating "markets for vulnerability." Some firms have attempted to create market-based mechanisms for vulnerability information. Firms such as iDefense (www.idefense.com) will pay identifiers for vulnerability information, which they provide exclusively to their clients.

Such arrangements have complex implications for overall security. Clearly, by providing incentives to find vulnerability, such firms increase the "supply" of vulnerability. Rescorla has argued that the likelihood of these vulnerabilities being rediscovered is small, so such incentives are socially wasteful.[7] (In terms of the model, if $F(s)$ is small enough, discovering vulnerabilities is socially wasteful.) However, this mechanism can have benefits if it spurs whitehats, increasing competition for blackhats.[11] Increasing competition can help keep blackhats from discovering vulnerabilities before whitehats, which can be socially beneficial. However, offsetting this benefit is the possibility that a for-profit intermediary can publicly release information about the vulnerability once its own clients are secured, leaving nonclients defenseless. This increases the benefit of becoming a client but reduces overall social welfare. Having a not-for-profit intermediary, such as CERT, is clearly superior,[11] indicating that vulnerability disclosure ought to remain in the nonprofit sector and requires public subsidies.

The theoretical framework we discuss highlights the complex set of issues surrounding how, when, and to whom vulnerability information should be disclosed. Our findings are suggestive rather than conclusive and point to several areas that require additional research. The framework needs to be enriched and extended to analyze issues such as interactions between competing vendors. However, even the rather simple framework highlights the need for collecting new data types. Although challenging, the task is feasible, and we're continuing to study this issue. □

Acknowledgments

The authors acknowledge financial support from Cylab and the Software Industry Center at Carnegie Mellon University.

References

1. "Symantec Internet Security Threat Report," *Symantec*, 2003; www.symantec.com/press/2003/n030203.html.
2. R. Farrow, "The Pros and Cons of Posting Vulnerability," *The Network Magazine,* 2000; www.networkmagazine.com/article/NMG20001003S0001.
3. E. Levy, "Full Disclosure Is a Necessary Evil," *SecurityFocus.com*, 2001; www.securityfocus.com/news/238.

4. E. Preston and J. Lofton, "Computer Security Publications: Information Economics, Shifting Liability, and the First Amendment," *Whittier Law Rev.*, vol. 24, 2002, pp. 71–142.
5. A. Arora, R. Telang, and H. Xu, "An Economic Model of Software Vulnerability Disclosure," *Proc. 3rd Workshop Economics and Information Security*, 2004; www.dtc.umn.edu/weis2004/.
6. E. Rescorla, " Security Holes ... Who Cares?" *Proc. 12th Usenix Security Conf.*, Usenix Assoc., 2003.
7. E. Rescorla, "Is Finding Security Holes a Good Idea?" *Proc. 3rd Workshop Economics and Information Security*, 2004; www.dtc.umn.edu/weis2004/.
8. A. Arora et al., *How Quickly Do They Patch? An Empirical Analysis of Vendor Response to Vulnerability Disclosure*, working paper, Carnegie Mellon Univ., 2004.
9. N. Kiefer, "Economic Duration Data and Hazard Functions," *J. Economic Literature*, vol. 26, no. 2, 1988, pp. 646–679.
10. A. Arora et al., *Impact of Patches and Software Vulnerability Information on Frequency of Security Attacks—An Empirical Analysis*, working paper, Carnegie Mellon Univ., 2004.
11. K. Kannan and R. Telang, "An Economic Analysis of Market for Software Vulnerabilities," *Proc. 3rd Workshop Economics and Information Security*, 2004; www.dtc.umn.edu/weis2004/.

***Ashish Arora** is a professor of economics and public policy and, by courtesy, computer science at Carnegie Mellon University, and the codirector of CMU's Software Industry Center (SWIC). His research focuses on the economics of technological change, intellectual property rights, and the economics of information security. Arora received a PhD in economics from Stanford University. Contact him at ashish@andrew.cmu.edu.*

***Rahul Telang** is an assistant professor of information systems at Carnegie Mellon University's H. John Heinz III School of Public Policy and Management. His research interests include economics of information security and consumers' use of new technologies such as search engines and peer-to-peer networks. Telang received a PhD in information systems from CMU. Contact him at rtelang@andrew.cmu.edu.*