Economics of Software Vulnerability Disclosure

Ashish Arora, Rahul Telang

H J Heinz III School of Public Policy and Management
Carnegie Mellon University

1. Introduction

Information security breaches frequently exploit software flaws or vulnerabilities.[1] Anecdotal evidence suggests that losses from such cyber-attacks can run in the millions.[2] There is considerable debate and disagreement about how vulnerabilities should be disclosed to the public. In this paper, we sketch a theoretical framework, which we use to identify the key data elements needed to develop a sensible way of handling vulnerability disclosure. Further, we briefly describe the results of the results of the empirical analyses of two data sets: vendor response to disclosure, and attack data from honeypots, which are useful for understanding how attackers respond to disclosure.

2. Vulnerability Disclosure

The CERT/CC (Center for Emergency Response Team/ Coordination Center), when informed of a vulnerability, will typically wait for the concerned vendor or vendors to provide patches or workarounds before issuing public "advisories" warning software users[3]. Unless the vendor response is "inordinately" delayed, CERT does not publicly disclose the vulnerability without a patch. However, many in the community believe that full and immediate disclosure is best. Full-disclosure mailing lists, such as "Bugtraq", were created in the late 1990s.[4]

[1] The shutting down of the eBay and Yahoo! websites due to hacker attacks and the Code Red virus, which affected more than 300,000 computers are just two well known examples where software defects were exploited. A recent report (Symantec, 2003) documents 2,524 vulnerabilities discovered in 2002, affecting over 2000 distinct products, an 81.5% increase over 2001. The CERT/CC (Computer Emergency Response Team / Coordination Center) received over 4000 reports of vulnerabilities in the year 2002 alone, associated with more than 82,000 incidents involving various cyber attacks.

[2] For example, CSI (Computer Security Institute) and FBI estimated that the cost per organization across all types of breaches was around $ 1 million in year 2000.

[3] The advisories contains among other items, technical information about a vulnerability and the patch information which users can download to protect their systems against potential exploits.

[4] Unlike CERT advisories, Bugtraq disclosures also frequently contain details of the vulnerabilities. We focus here

Proponents of full disclosure claim that the threat of instant disclosure increases public awareness, puts pressure on the vendors to issue high quality patches quickly, and improves the quality of software over time. But many believe that disclosure of vulnerabilities, especially without a good patch is dangerous. Richard Clarke, President Bush's former special advisor for cyber space security, criticizing full disclosure said: “It is irresponsible and … damaging to release information before the patch is out.” [5] Scott Culp, manager of Microsoft Security Response Center, described full disclosure as “information anarchy”. The public policy problem is important but there is little extant research to guide policy.

3. Literature Review

Only a few papers have analyzed economic issues related to problems in the information security. Arbaugh et. al (2000) provide a vulnerability life cycle and show that many exploits occur after the patch has been released. Schechter (2002) argues that encouraging competition among testers to discover vulnerabilities can improve quality. Kannan and Telang (2004) 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. Preston and Lofton (2002) provide an overview of the current state of law and regulation for vulnerability disclosure and forcefully argue against any restriction on publication of such information. Rescorla (2004) argues that finding a security hole, let alone disclosing it, is socially wasteful because the probability that the hacker would find it is very small. Gordon et al. (2002) discuss how the economic issues related to information sharing in Information Sharing & Analysis Centers (ISACs) are similar to those in trade associations. Gordon et al. (2003) provide a theoretical analysis of how information sharing mediates the impact of security investment on expected security costs.

4. Framework and Model

The outcome of any disclosure policy depends upon how three major sets of participants respond: software vendors, users, and white and black hat hackers. Disclosure policies affect

on “when” rather than “how much” information is disclosed, though there are obvious parallels between the two.
[5] Refer to: http://www.blackhat.com/html/bh-usa-02/bh-usa-02-speakers.html#Richard%20Clarke. For details on this debate see (Farrow 2000; Elias 2001; Preston and Lofton 2002)

each participant differently. Below we present a theoretical model, based on Arora, Telang and Xu (2004), to analyze these impacts.

There are three major participants in our model: a vendor, user and attacker. In figure 1, the product is released and used by users at time 0.[6] Since disclosure policy is mostly irrelevant if a black hat discovers the vulnerability, we focus on the case where the vulnerability is discovered by a white hat, who discovers the vulnerability at t_0.

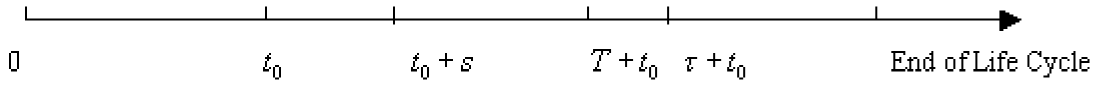

Figure 1. Software Life Cycle

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 vulnerability information is shared with only the vendor. Once time *T* elapses, the information is disclosed to the public irrespective of the availability of patch. Instant disclosure policy means $T = 0$ while secrecy policy implies a $T = \infty$. In terms of figure 1, a disclosure policy *T* requires that this vulnerability is disclosed time $T + t_0$.[7]

Vendors provide a patch for this vulnerability at a calendar time $\tau + t_0$, possibly after disclosure. Note that τ, T and s are measured from the calendar time t_0, which is the time when the vulnerability is first discovered by a white hat. 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, attackers are assumed to be able to exploit the vulnerability without any further delay.[8] The expected user loss is $L_e(\tau, T : X)$, a function of the disclosure policy *T* and the time window for patching, τ. Users suffer loss when black-hats discover the vulnerability, on their own or through disclosure, before patch is available.

[6] We ignore the diffusion of the product and assume that all users start using the product at time '0'.
[7] The goal is to study the socially optimal tradeoffs in when to disclose. 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 is no interesting policy question. Formally, this is as if the official disclosure time were zero.
[8] According to a recent report (Symantec, 2003), approximately 60% of the documented vulnerabilities can be exploited almost instantly either because exploit code is widely available or because no exploit tool is needed. Thus

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, since the number of black-hats who learn about the vulnerability and the chances of an exploit tool being developed will increase with exposure time. It follows that expected loss, $L_e(\tau, T)$, will critically depend on when the patch is made available (τ) and when the vulnerability is disclosed (T). Consider the following two cases:

C1: Patch is released before *T;*

C2: Patch is released after *T*.

When patch is released before disclosure time (C1), users suffer loss only if attackers finds the vulnerability on its own and prior to the patch. In Figure 1, 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, if the patch is released after T (C2), attacker can find the vulnerability on its own exploit it for $\tau-s$.[9] Alternatively, attacker learns about the vulnerability when it is disclosed at T and exploits it till the patch is made available, for $\tau-T$.

To capture the uncertainty about when a vulnerability will also be discovered by an attacker, we assume that the time that attacker finds the vulnerability (s) , conditional on a white hat discovering it at t_0, is stochastic, with a distribution $F(s;\ t_0)$. Therefore, the probability that attacker does not find it within period T is simply $1-F(T:t_0)$, where t_0 is the calendar time when the vulnerability was first discovered. We assume that $F(s:t_0)$ increases with t_0 because as attackers accumulate experience and knowledge about the software, they are more likely to find the vulnerability.

Thus, the expected user loss can be written as follows:

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

[9] As discussed later, if all users do not patch, additional damage may result after the patch is released.

The first part of the expected user loss function is when patch is released before T but attacker finds the vulnerability at a time s ($s < \tau$) and exploit it for the duration $\tau - s$. The second part is when patch is released after T, and attacker can either find it either before T and attack for $\tau - s$ or find about it at time T when it is disclosed by social planner and attack for duration $\tau - T$.

Given a disclosure policy T, the vendor makes decision on allocating its resources in making the patch available. The vendor's objective function (modeled here as a cost function to be minimized) has two terms. The first term is the cost of developing the patch, $C(\tau)$. Recall that τ is the time taken to develop a patch. All else held constant, the quicker the patch, the higher are the development costs, i.e. C is a decreasing function of τ.

The second cost is a proportion of user loss that vendor internalizes (via a loss in reputation, loss of future sales), represented here by λ. Currently, vendors do not face any legal liability from losses arising due to vulnerabilities in their products but this may change in the future. When $\lambda = 1$ the vendor internalizes the entire loss to users and therefore 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, the vendor is assumed to make a one-time, committed decision on when to patch. As well patching time here is deterministic and quality of patch is assumed fixed. Rescorla (2003) provides a case study that shows that users do not apply patches immediately after they are available. This framework can be extended (see Arora, Telang and Xu, 2004) to the case where not all users install patch right away upon the release of the patch (only a fraction $p(x)$ of users apply the patch at time x where $x > \tau$ and $p' > 0$) and where the probability of patching depends upon the quality of the patch, which the vendor can choose. Other extensions include explicitly allowing for users to protect themselves even absent a patch. Further extension may consider allowing vendor to make real-time decision (from time to time) on when to develop or release an already-developed patch in response to changes in the environment (e.g., news that black hats have exploited the vulnerability).

In addition, this model ignores two important aspects of the debate on disclosure. Full disclosure proponents argue that disclosure puts pressure on the vendors to ultimately improve the quality of their software. Our model deals more with the ex-post release of patches than ex-ante quality of the software. In addition, disclosure may educate programmers or network administrators on new classes of bugs and improves the overall level of security.

However, the model captures important elements of the tradeoff. It captures the cost of rushed patching to vendors via $C(\tau)$. It can be used to analyze how disclosure policies affect vendor behavior, and the factors, such as patching costs, likelihood and severity of user loss, vulnerability characteristics and the legal liability regime (if it were to be in place) and vendor characteristics (for example, open source vs. closed source) condition their response. Most importantly, given the behavior of black hats and users, as captured by $F(s)$ and $p(x)$, we can ask: 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.

The key implications of our framework are

a) Vendors respond to quicker disclosure by quicker patches. Therefore, instant disclosure would force vendors to issue patches faster though it need not be an optimal policy.
b) If users do not patch their systems quickly then an 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.
c) If vendors make the decision to issue a patch on real time basis (rather than making a commitment to patch earlier) then an optimal policy is to never disclose the vulnerability or issue a patch until the hacker finds it. Once the hacker finds it, vendor can issue a patch immediately.
d) For a new or recently release product, setting a larger disclosure window is useful because since the product is not extensively understood, the exploit tools are not widely available. Moreover if $F(s)$ (probability of hackers finding it conditional of a white hat has found it) were

to be very small, as Rescorla (2004) argues, then optimal policy is to never disclose the vulnerability.

e) As the internalization factor (or legal liability if possible) increase, vendors are more aggressive in patching their systems. Similarly, the optimal disclosure window also reduces.

In the following two sections, we will briefly describe two empirical studies which follow from our model. In the first study, we empirically investigate the time it takes for vendors to patch vulnerability in their systems and test whether this time systematically differs when the vulnerability was disclosed as opposed to when it is still secret. In the second study, we empirically examine the patterns over time of attacks experienced by a host when the vulnerability is secret or, disclosed, or patched.

5. Empirical Studies

Study 1: Vendor's Response to Disclosure Policy

Our framework above provides us an understanding of how vendors might behave under different disclosure policy regimes. In particular, we found that instant disclosure without a patch may increase customer loss. If the vendors internalize some of the costs then they may have incentive to provide a patch early. The exact timing of patch also depends on the cost of patching. If the cost of patching were trivially small then for some vulnerabilities we may not see significant differences between the two policies. While we do not observe the cost of patching, we observe the vulnerability characteristics and vendor type (public firm, open source vendor, size of the vendor etc).

Arora et. al. (2004 A) collected the data from two sources, CERT/CC and security focus's "Bugtraq" mailing list. As noted earlier, unless the vendor response is "inordinately" delayed, CERT does not publicly disclose the vulnerability without a patch. On the other hand, many vulnerabilities are posted without the patch (and sometimes with the exploit code) on Bugtraq. As a first approximation, Bugtraq corresponds to instant disclosure while CERT typically provides at least 45 days for the vendor to provide a patch.

We collected 504 observations from both sources and examine how the decision to patch, and how soon to patch changes with the policy, controlling for various vulnerability and the vendor characteristics. The data presents methodological challenges as CERT characterizes the effect of vulnerability in many vendors as “unknown” even though they could be potentially vulnerable. First, we use a joint sample from CERT and Bugtraq to estimate the true number of “vulnerable” in the “unknown” category. Then we estimate the model first using a probit framework to examine how the probability of vulnerabilities getting patched differs between two policy regimes and then using proportional hazard model (Cox PHM – Kiefer 1997) to understand the speed of patching between two regimes.

The average patching time for CERT (N = 186) was 242 days and 390 days for Bugtraq (N = 318). Approximately 60% of vulnerabilities were patched on Bugtraq while 77% were patched on CERT. When we control of other factors, we find that both the probability of patching and the speed of patching is approximately same in both regimes. This is an interesting and surprising finding because our results do not provide support for the claim full disclosure pushes the vendors into developing patches more quickly. One limitation of our research was that we did not control for the quality of the patch. Subject to this qualification, our results suggest if there are benefits of early disclosure, they must lie in allowing users to protect themselves without a patch, and perhaps in the incentives for vendors to develop more secure products in the first place.[10]

<u>Study 2: Attackers’ Behavior to Disclosure</u>

In the second study, we empirically explore the impact of vulnerability information disclosure and availability of patches on number of attacks seeking to exploit the vulnerability. Arora et. al (2004 B) collected data mainly from 14 “honeypots” run on different locations and operating on different operating environments - Linux, Solaris, OpenBSD and Window for several weeks in a year. Unlike real networks, where distinguishing between an attack and a legitimate traffic is not always possible, honeynets provides an easy way to detect attacks as honeypots by definition do not have legitimate network traffic (Stuart and Smith, 2000).

[10] We also find that open source vendors are quicker to patch than closed source vendors and that more severe vulnerabilities are patched faster. We also find that more recent vulnerabilities (after 09/11/2001) are patched more often and faster.

We created our key variable – the number of attacks targeting a vulnerability – by matching attack data with attack traffic signatures obtained from publicly available sources such as www.whitehats.com and http://www.snort.org/cgi-bin/done.cgi. We selected vulnerabilities at random from the Common Exposures and Vulnerability (CVE) ICAT database, a public database on software vulnerabilities. We classify vulnerabilities as either *secret* (all secret vulnerabilities were eventually disclosed), *published* or *patched*. A *secret* vulnerability is one that is neither published nor patched, a *published* vulnerability is published but not patched, and a *patched vulnerability* is both published and patched. A given vulnerability could go from being *secret* to *published* to *patched*.

The data so assembled consists of 2952 observations over 9 weeks from Nov. 2002 to Dec 2003 for 328 different vulnerabilities. Of 328 vulnerabilities, 77 vulnerabilities had no patches. 160 vulnerabilities were made public on the same day when a patch fixing them was also released and 76 vulnerabilities were patched afterwards. Only 44 vulnerabilities were exploited in our data. We found that on an average secret vulnerabilities (N = 24) received 0.32 attacks per host/per day, while published vulnerabilities (N = 77) attracted 5.45 attacks per host/per day and patched vulnerabilities (N = 233) attracted 2.5 attacks per host/per day.

We find that publishing of vulnerabilities attracts attacks, which decline with time. The release of the patch also attracts attacks initially which then decline with time. When averaged over time, the results imply that disclosure results in more attacks but patching reduces attacks. *Secret* vulnerabilities suffer the least number of attacks. While it may be tempting to interpret this as evidence in favor of secrecy, note that if a secret vulnerability suffered a lot of attacks, it would quickly become public and thus the average number of attacks over a time period for vulnerability that remains secret over the period must by definition be small. Moreover, the study analyzes observed attacks and not the economic loss per se; though secret vulnerabilities are naturally less frequently attacked, such attacks may result in greater damage. The finding that the release of a patch 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 will not patch promptly enough.

6. Discussion

In this paper, we first present a normative model for choosing a socially optimal disclosure policy and then present two empirical studies which provide suggestive evidence on two key elements of the model, namely vendor and attacker response to disclosure. Our empirical results suggest that full disclosure provides at best a very modest incentive for vendors to develop patches more quickly. Our results are also consistent with other findings which indicate that a significant fraction of users may not install patches promptly. Further, since disclosure on Bugtraq may take place without a patch and in some cases, even exploit code may be provided, our results do not support full and instant disclosure as optimal policy. [11] Put differently, our results imply that proponents of full and instant disclosure can only be socially optimal if either it prompts vendors to develop more secure products; or if it enables users to protect themselves in other ways, which include workarounds or additional perimeter defenses. Both of these questions require additional empirical research.

Our discussion raises the fundamental question of who "owns" the vulnerability information. Some authors have argued for creating "markets for vulnerability". There are also attempts by some firms to create a market based mechanism for vulnerability information. Firms like iDefence are willing to pay money to identifiers for vulnerability information which they provide exclusively to their clients.

The implications of such arrangements for overall security are complex. Clearly, by providing incentives to find vulnerability, such firms increase the "supply" of vulnerability. Insofar as the likelihood of these vulnerabilities being rediscovered is small, as Rescorla (2004) has argued, this is socially wasteful.[12] However, Kannan and Telang (2004) show that this mechanism can have benefits if it spurs white-hats and thus creates greater competition for black-hats. Insofar as this greater competition reduces the rents to black-hats from discovering vulnerabilities, market based arrangements 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 non-clients defenseless. This increase the benefit of becoming a

[11] Some argue that CERT does not provide enough information to users even when it discloses a vulnerability. Therefore, a forum like Bugtraq is useful because it provides detail information regarding vulnerability. Our research does not consider the issue of "quality" of disclosure.

[12] In terms of the model, if $F(s)$ is small enough, discovering vulnerabilities is socially wasteful.

client but reduces overall social welfare. Kannan and Telang show that having a not-for-profit intermediary, such as CERT (now US-CERT) is clearly superior, which makes recent reports of the privatization of vulnerability disclosure by US-CERT even more troubling.

How, to whom and when vulnerability information should be disclosed raises a complex set of issues that require systematic analysis and the collection of new types of data. Though challenging, the task is feasible. This paper has summarized the results from a variety of ongoing studies we are engaged in with other colleagues. Our findings are suggestive rather than conclusive and point to a number of different areas that require additional research.

References:

Arbaugh, W.A., Fithen, W. L. & McHugh, J. (2000), "Windows of Vulnerability: A Case Study Analysis", *(IEEE) Computer*.

Arora, A., R. Telang, and H. Xu (2004, May). An Economic Model of Software Vulnerability Disclosure. *The Third Workshop on Economics and Information Security*. Minneapolis MN.

Arora, A., Krishnan R., Telang R., and Y. Yang (2004A). "How quickly do they Patch? An Empirical Analysis of Vendor Response to Vulnerability Disclosure" *Working paper*, Carnegie Mellon University.

Arora A., Nandkumar A., Krishnan R. and Telang R (2004B), "Impact of Patches and Software Vulnerability Information on Frequency of Security Attacks - An Empirical Analysis", *Working paper*, Carnegie Mellon University.

Elias, L. (2001) "Full Disclosure is a necessary Evil" *SecurityFocus.com* www.securityfocus.com/news/238.

Farrow, R. (2000), "The Pros and Cons of Posting Vulnerability" *The Network Magazine*.

Gordon, L. A., Loeb M.P., and Lucyshyn W. (2003) "Sharing Information on Computer Systems: An Economic Analysis" *Journal of Accounting and Public Policy 22*(6), 461–485.

Kannan K. and Telang R. (2004), "An Economic Analysis of Market for Software Vulnerabilities", *The Third Workshop on Economics and Information Security*. Minneapolis MN.

Kiefer, N. (1988), "Economic Duration Data and Hazard Functions", *Journal of Economic Literature*, 26(2), 646-679.

Preston, E. and Lofton, J. (2002), "Computer security publications: information economics, shifting liability and the first amendment", *24 Whittier Law Review, 71-142.*

Rescorla, E. (2003), “ Security holes... Who cares?”, *Proceedings of the 12th USENIX Security Conference*, August.

Rescorla, E. (2004), “Is finding security holes a good idea?”, *The Third Workshop on Economics and Information Security*. Minneapolis MN.

Schechter, S.E. & Smith, M.D. (2003). How Much Security is Enough to Stop a Thief?, *The Seventh International Financial Cryptography Conference, Gosier, Guadeloupe, January*.

Symantec Inc. (2003), “Symantec Internet Security Threat Report”. http://www.symantec.com