Impact of Vulnerability Disclosure and Patch Availability - An Empirical Analysis

Ashish Arora, Ramayya Krishnan, Anand Nandkumar , Rahul Telang and Yubao Yang

H. John Heinz III School of Public Policy and Management
Carnegie Mellon University, Pittsburgh PA 15213

Email: {ashish; rk2x; anandn; rtelang; yubaoy}@andrew.cmu.edu

April 2004

Abstract

Vulnerability disclosure is an area of public policy that has been subject to considerable debate, particularly between proponents of full and instant disclosure, and those of limited or no disclosure. This paper is an attempt to empirically test the impact of vulnerability information disclosure and availability of patches on attackers' tendency to exploit vulnerabilities on one hand and on the vendors' tendency to release patches on the other. Our results suggest that while vendors are quick to respond to instant disclosure, vulnerability disclosure also increases the frequency of attacks. However, the frequency of attacks decreases over time. We also find that open source vendors patch more quickly than closed source vendors and that large vendors are more responsive.

Keywords: Software Vulnerability, Full disclosure policy, attackers, patching behavior

1. Introduction

There is a contentious ongoing debate about how vulnerability information should be made public. While information about vulnerabilities enables some users to take precautions that prevent or reduce cyber security breaches, vulnerability information, especially when not accompanied by patches or workarounds, can benefit attackers more than users. There are many sources that report vulnerability information, ranging from federally funded quasi-government organizations like CERT/CC (the CERT® Coordination Center, the first computer security incident response team.) privately owned consulting companies. Traditionally, CERT has been a key player in the domain of vulnerability disclosure. A typical sequence of events in case of CERT is as follows. A *benign identifier* reports the vulnerability to CERT, which then contacts the vendors involved and provides them a certain time window to patch the vulnerability. After that time, CERT sends out public "advisories" warning users about the vulnerability. The advisories include links to patches if available and provide enough technical information about the vulnerability to enable users to take protective action. However, many identifiers also use public forums, such as the "Bugtraq" mailing list. Here, some identifiers make public all information including technical details of a vulnerability (sometimes also including the exploit code) even if the vendor has not released a patch.[1]

While the proponents of instant disclosure claim that disclosing vulnerability information provides an impetus to the vendor to release patches early, the proponents of secrecy claim that *instant disclosure* leaves users defenseless against attackers who can exploit the disclosed vulnerability and therefore, are socially undesirable (see Elias 2001 and Farrow 2000)[2]. Gordon et al. [1999], while acknowledging the importance of these issues, point out the lack of hard evidence to assess the impact of various forms of

[1] This type of disclosure is known as *instantaneous disclosure*.
[2] See also the debate between Robert Graham and Bruce Schneier. http://www.robertgraham.com/diary/disclosure.html LAST CHECKED???

vulnerability disclosure. One major problem is lack of empirical evidence along a number of relevant dimensions. This paper is an attempt to empirically test the impact of vulnerability information disclosure and availability of patches on

(i) Number of attacks seeking to exploit the vulnerability
(ii) How promptly vendors release patches.

The empirical estimates on attacker's and vendors' behavior are crucial for our ability to formulate an optimal disclosure policy. If the vendors do not patch quickly to instant disclosure then such policies are clearly socially detrimental and should be strongly discouraged. But even if they do patch earlier, we need to know how the attacker's probability of attack changes with the disclosure, and with the patching. We investigate which types of vulnerabilities are more likely to be exploited by attackers, which need immediate user intervention and which are more likely to be patched by the vendor. We also examine if the large vendors and the open source vendors are more responsive.

It has been argued by many researchers that Information security is not a problem that technology alone can solve and that it should be treated as a risk management problem [Schneier 2002]. Increasingly investments in countermeasures are being viewed within a risk management paradigm, not altogether different from those used for other business losses. But there is little empirical data to guide businesses on how to make this risk return tradeoff (Gordon, Loeb and Sohail [2003], Hoo [2000]).

Using a game theoretic model Arora, Telang and Xu [2003] examine how disclosure affects vendor behavior, and the implications for when a social planner would optimally disclose a vulnerability. They find that although early disclosure could result in the vendor releasing a patch more quickly, it is not necessarily optimal. In general, they show that neither instantaneous disclosure nor secrecy policy is optimal. The optimal policy depends upon underlying factors such as how quickly vendors respond to disclosure by releasing patches, and upon how likely attackers are to find and exploit undisclosed or un-patched vulnerabilities. Our paper provides empirical evidence on these key underlying factors. Specifically, we investigate the impact of vulnerability disclosure on when vendors release patches, and of vulnerability disclosure and patching on the frequency of attacks seeking to exploit the vulnerability and the changes over time.

Our preliminary empirical results are as follows. First, vulnerability disclosure increases the number of attacks on hosts, while the availability of patches reduces the number of attacks on hosts. Interestingly, the results also indicate that keeping vulnerability information secret may results in fewer attacks. Second, older vulnerabilities are less likely to be exploited by attackers than newer vulnerabilities. Similarly, vulnerabilities for which patches were released earlier are exploited less that those for which patches are relatively new. Third, instantaneous disclosure policy increases the probability of vendors fixing vulnerabilities and vendors are also more likely to patch and patch faster under instantaneous disclosure. Finally, open source software vendors patch faster than close source vendors, and large vendors are more responsive to vulnerabilities disclosed in their products.

The reminder of this paper is organized as follows: section 2 focuses with the impact of vulnerability disclosure on attackers' behavior and section 3 focuses on the impact of vulnerability disclosure policy on the vendors' patching behavior. In both these sections, we provide details on our data collection and our analysis. We conclude with a summary of the results with a discussion in section 4.

2. Impact of vulnerability disclosure and Patch availability on attack frequency

This part of the paper deals with an empirical analysis of the impact of vulnerability disclosure and patch availability on the attackers' tendency to exploit the vulnerability. In this part, we use the term "attack" to include attempts made by an attacker to compromise a remote host. This part is organized as follows:

Section 2.1 provides details about the data sources used in this paper. Section 2.2 contains a description of the econometric models and empirical estimates.

Data

We acquired two types of data for the purposes of this paper – data on security incidents and data on vulnerabilities that resulted in the security incidents. The first part of data comes from the honeypots run by www.honeynet.org and its affiliated members. A honeypot is a system that emulates a computer that is connected to the Internet. These are typically used to capture extensive data on information security attacks and motives of attackers [Spitzner 2000]. 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].

The data consists of network traces from 14 honeypots operating on different operating environments – Linux, Solaris, OpenBSD and Windows – collected for several weeks over the course of a year. The honeypots were placed behind a firewall, with each honeypot having a separate IP address. The honeypots had no legitimate applications hosted on them. The honeypot data primarily consists of *tcpdump* traces of individual packets, both inbound and outbound. The data so captured consists of data on all the TCP/IP packets that entered or left any of the 14 honeypots along with the date and time, nature of the packet (payload), the source and destination addresses and also the source and destination processes. The data captured were stored in a secured remote database

Data from honeypots are a valuable resource because they do not face the usual biases due to selection in detection and in reporting, present in most field data. Therefore, it is easier to classify attacks and eliminate false positives. However, though providing many advantages, there are some limitations as well. First, an actual system will have legitimate traffic, so that the frequencies of attempted break-ins may systematically vary from those implied by the honeypot data. Further honeypots cannot provide insight into targeted attacks on an organization, nor for internal attacks that are mounted by employees with an organization. But despite these limitations, honeypots are a valuable data source, particularly in view of the paucity of reliable field data and the strong selection biases that such field data likely contain.

Extracting attack data

We created our key variable – the frequency of attacks targeting a vulnerability – by matching attack data with attack traffic signatures. Attack signatures are a set of rules that identify malicious packets and link them to specific vulnerabilities targeted. These signatures are based on packet payload, destination port and address, source port and address, packet sequence number, protocol or any combination of these. The attack signatures were acquired from publicly available source, specifically, Whitehats (www.whitehats.com) and Snort database (http://www.snort.org/cgi-bin/done.cgi). We implemented a custom parser based on WinTcpdump library and matched the tcpdump traffic from honeypots with attack signatures and collated them with vulnerabilities. This provides us with a count of the number of attempts to exploit a specific vulnerability, henceforth called *the number o f attacks,* over a given period[3]

Vulnerability data

We selected 308 unique vulnerabilities at random from the from the Common Exposures and Vulnerability (CVE) ICAT database. The CVE ICAT database is a publicly available database that contains information about software vulnerabilities. The database aggregates information about software vulnerabilities from other public forums like CERT, Bugtraq or ISS. This database currently contains information on about 6000 vulnerabilities disclosed in various public forums from 1989 to 2004. Each

[3] Each tcpdump data file consists of tcp logs accumulated from 12:01 AM of the start day till 12:00 AM of the end day during a period. Each period of observation typically contains about 5 days of observation.

vulnerability has a unique identifier known as CVE-ID and is further characterized by other descriptors like date of publication, severity type, vulnerability type[4] and vendor whose software is vulnerable. We augmented vulnerability information with information on patches and exploit code[5]. While information on patches fixing vulnerabilities was acquired from the web site of different vendors, data about the availability of exploit code was acquired from different publicly available forums like Bugtraq (www.online.securityfocus.com), mailing list ARChives at AIMS (http://marc.theaimsgroup. com), ISS (www.Xforce.ISS.net) and Packetstorm (www.packetstorm.org).

The data so assembled consists of 2772 observations over 9 weeks from Nov. 2002 to Dec 2003 for 308 different vulnerabilities. Of 308 vulnerabilities, 73 vulnerabilities had no patches[6] released by the vendor. About 160 vulnerabilities were made public on the same day[7] when a patch fixing them was also released. About 75 vulnerabilities were patched before information about the vulnerability was made public.

We classify vulnerabilities as either *secret*[8], *published*[9] or *patched*. A *secret* vulnerability is one that is neither patched nor published, a *published* vulnerability is published but not patched, and a *patched vulnerability* is both published and patched. In this paper, publishing a vulnerability is interpreted as the act of making information about a vulnerability public through public forums like CERT, Bugtraq etc. or by the vendor on its website. Tables 1 through 3 provide descriptive statistics of the sample.

Table 1 – Period wise breakup of vulnerabilities

Period	*Number of days of observation*	Patched	Published	Secret
Period 1Nov 2002	5	206	78	24
Period 2 (Jan 2003)	6	210	77	21
Period 3 (Jan 2003)	5	210	77	21
Period 4 (Jan 2003)	7	210	77	21
Period 5 (Mar 2003)	7	212	78	18
Period 6 (May 2003)	7	223	77	8
Period 7 (Sep 2003)	7	231	75	2
Period 8 (Nov 2003)	7	231	75	2
Period 9 (Dec 2003)	7	233	75	0

[4] Severity type consists of identifiers for how severe the vulnerability is based on the possible damage that could result on the attacked host. Severity type includes security protection, confidentiality, integrity and availability. Vulnerability type denotes the technical characteristics of the vulnerability such as input validation error, boundary condition error, buffer overflow, access validation error, exceptional condition, environmental error, configuration error, race condition and other vulnerability

[5]Exploit code also includes cases where no actual code is provided but where explanations on how to exploit the vulnerability are available.

[6] Vulnerabilities that were never patched would take a value of zero for the elapsed *patch days* but the dummy variable that denotes "not patched" vulnerabilities would take on a value of 1.

[7] Vulnerabilities that have been patched before they were published were deemed to have been patched and published on the same day.

[8] Vulnerabilities for which *elapsed patch days* and *elapsed publish days* are both less than zero are *secret* vulnerabilities. All secret vulnerabilities in our sample were eventually disclosed.

[9] These are vulnerabilities that have been published but not yet patched and would contain a positive value for *elapsed publish days* and negative values for *elapsed patched days*.

Table 2 – Descriptive statistics

	Secret	*Published*	*patched*
Average number of attacks per host per day (attacks)	0.3145	5.45	2.50
Std. Deviation of average number of attacks	0.351	20.33	5.01
No of exploited vulnerabilities	22	59	57
Average age of exploited vulnerabilities (days from publication)[10]	-231	899.61	868.92
Minimum age of exploited vulnerabilities (days from publication)	-364	95	10
Maximum age of exploited vulnerabilities (days from publication)	-116	2144	2022
Average age of patches for exploited vulnerabilities (days from patch)[11]	233	-70.8	771.95
No of unexploited vulnerabilities	95	630	1909
Average age of unexploited vulnerabilities (days from publication)	-109	1182.88	1092.80
Minimum age of unexploited vulnerabilities (days from publication)	-363	27	4
Maximum age of unexploited vulnerabilities (days from publication)	0	3278	5548
Average age of patches for unexploited vulnerabilities (days from patch)	-116	-3.96	991

Table 3 - Vulnerability by type

	Number of vulns.	*Number of observations*
Vulnerability only affecting windows hosts	53	477
Vulnerability only affecting Linux hosts	17	153
Vulnerability only affecting Solaris hosts	11	99
Vulnerability only affecting all UNIX hosts	25	225
Vulnerability only affecting all hosts	10	90
Other vulnerabilities+	192	1728

+Other vulnerabilities are those that not operating system vulnerabilities but vulnerabilities that pertain to application software that reside on an operating system that affects host, such as FTP client software vulnerability.

2.1. Empirical estimates

In the first part of this section we examine the average effect of patches and vulnerability disclosure. In the second part, we will examine the effect of elapsed days from the availability of patches and the effect of elapsed days from publishing on the attack frequency.

Average effect of patching and publishing: Results from Non Parametric Analysis

To understand the effect of patching we compare the differences in attacks per host per period between *patched* vulnerabilities and *published* vulnerabilities. Column (5) of table 4 indicates that availability of patches increases attacks on hosts at the rate of 0.02 attacks per day. Similarly, a comparison between

[10] The age of a vulnerability is measured as the difference, in days, between the date of publication of vulnerability and the first day of the observation period, and takes negative values if the date of observation precedes publication.
[11] The age of patch is measured as the difference, in days, between the date of release of patch by vendor and the first day of the observation period, taking negative values if the date of observation precedes the release of the patch.

published vulnerabilities (column 7) and *secret* vulnerabilities (column 2) suggests that disclosure of vulnerability information too increases the number of attacks on hosts at the rate of 0.02 attacks per day. The "average effects" reported in table 4 are the difference between the change over time in attacks between patched and un-patched, and between secret and published vulnerabilities. These estimates implicitly control for vulnerability characteristics by looking at changes over time within a vulnerability. However, these results do not explicitly control for time and location effects, for which we turn to regression results.

Vulnerability characteristics vs. attack "fixed effects": Regression results

There are two ways of controlling for vulnerability characteristics in a regression framework: Controlling for vulnerability characteristics, and including vulnerability fixed effects. We show the results from both. We regress the number of attacks on *patched*, *secret* and published. We can either estimate a fixed effect model with vulnerability specific dummies or we can include vulnerability specific details. If A_{it} denotes the number of attacks of vulnerability type *i* on during period *t*, then we estimate,

$$A_{it} = \alpha_i + \delta_0 + \delta_t timedummies_t + \delta_2 location_t + \beta_1 \sec ret_{it} + \beta_2 published_{it} + e_{it} \qquad (1)$$

where α_i denotes the vulnerability fixed effects, $timedummies_t$ are dummy variables indexing time periods to control of time effects, and $location_i$ is the dummy variable that denotes the location of honeypots from which data was gathered.. We set $secret_i=1$, if vulnerability type *i* is *secret* and 0 otherwise, and similarly $published_{it} = 1$, if the vulnerability has been *published* and 0 otherwise.

The average number of attacks on vulnerabilities that have been patched and published is

$$E(\overline{A_{it}} \mid published_{it} = 0, \text{secret}_{it} = 0) = \hat{\alpha}_i + \hat{\delta}_0 + \hat{\delta}_t + \hat{\delta}_2 \qquad (2)$$

The average number of attacks on the vulnerabilities that have not been patched but published is

$$E(\overline{A_{it}} \mid published_{it} = 1, \text{secret}_{it} = 0) = \hat{\alpha}_i + \hat{\delta}_0 + \hat{\delta}_t + \hat{\delta}_2 + \hat{\beta}_2 \qquad (3)$$

The average effect of patching is

$$[E(\overline{A_{it}} \mid published_{it} = 0, \text{secret}_{it} = 0) - E(\overline{A_{it}} \mid published_{it} = 1, \sec ret_{it} = 0)] = -\hat{\beta}_2 \qquad (4)$$

The average number of attacks on vulnerabilities that are "not patched" and "not published" is

$$E(\overline{A_{it}} \mid published_{it} = 0, \text{secret}_{it} = 1) = \hat{\alpha}_i + \hat{\delta}_0 + \hat{\delta}_t + \hat{\delta}_2 + \hat{\beta}_1 \qquad (5)$$

Using (3) and (5) the average effect of "publishing" is

$$[E(\overline{A_{it}} \mid published_{it} = 1, \text{secret}_{it} = 0) - E(\overline{A_{it}} \mid published_{it} = 0, \sec ret_{it} = 1)] = \hat{\beta}_2 - \hat{\beta}_1 \qquad (6)$$

Table 5 provides the estimated result using both fixed effects (column 2) and vulnerability specific characteristics (column 3). Since the results are different in both scenarios, it suggests that the available vulnerability characteristics may be insufficient to account for all the heterogeneity across vulnerabilities. We focus henceforth on the results using vulnerability fixed effects.

Table 4: Difference in means of average number of attacks per host per day

	Patched (1)	Difference in (1) between periods (2)	Published (3)	Difference in (3) between periods (4)	Effect of patching = (2) – (4) (5)	Secret (6)	Difference in (6) between periods (7)	Effect of publishing Col.(7) – col.(2) (8)
Period 1 (Nov 2002)	0.05 (0.002)	-	0.22 (0.11)	-	-	0.004 (0.001)	-	-
Period 2 (Jan 2003)	0.06 (0.001)	0.01 (0.003)	0.53 (0.03)	0.31 (0.14)	-0.30 (0.14)	0.06 (0.005)	0.056 (0.01)	0.05 (0.01)
Period 3 (Jan 2003)	0.06 (0.002)	0	2.29 (0.23)	1.76 (0.26)	-1.76 (0.26)	0.04 (0.004)	-0.02 (0.01)	-0.02 (0.01)
Period 4 (Jan 2003)	0.07 (0.004)	0.01 (0.01)	0.18 (0.17)	-2.11 (0.40)	2.10 (0.41)	0.10 (0.009)	0.06 (0.01)	0.05 (0.02)
Period 5 (Mar 2003)	0.11 (0.000)	0.04 (0.004)	0.32 (0.010)	0.14 (0.18)	-0.10 (0.18)	0.06 (0.004)	-0.04 (0.01)	0
Period 6 (May 2003)	0.13 (0.002)	0.02 (0.002)	0.37 (0.014)	0.05 (0.02)	-0.03 (0.02)	0.11 (0.017)	0.05 (0.02)	0.03 (0.02)
Period 7 (Sep 2003)	0.01 (0.000)	-0.12 (0.002)	0.01 (0.007)	-0.36 (0.02)	0.24 (0.02)	0	-0.11 (0.02)	0.01 (0.02)
Period 8 (Nov 2003)	0.01 (0.005)	0 (0.005)	0.02 (0.002)	0.01 (0.01)	-0.01 (0.01)	0	0	0
Period 9 (Dec 2003)	0.01 (0.004)	0	0.001 (0.001)	-0.02 (0.003)	0.02 (0.003)	0	0	0
Average effect					**0.02 (0.13)**			**0.02 (0.003)**

Notes: Standard errors in parentheses.

The results in table 5 qualitatively similar to the differences-in-means estimates. From table 5, the coefficient of $\hat{\beta}_2$ is -0.17 while the coefficient of $\hat{\beta}_1$ is -0.20. Both the estimates are statistically significant at 10%. Thus, the availability of patches is associated with an increase of about 0.17 attacks per host per day, while publishing vulnerabilities is associated with an increase of about 0.03 attacks per host per day. Clearly, *patching* information benefits attackers as well: If users do not patch right away.

Since not all users may patch quickly, attackers have an incentive to try out attacks even for those vulnerabilities that have been patched. The result also suggests that, on average, both *published* vulnerabilities and *patched* vulnerabilities are likely to be exploited more than *secret* vulnerabilities.

Table 5: Impact of Patching and Publishing (OLS estimates)

Variables	Specification 1: vulnerability characteristics+		Specification 2: vulnerability fixed effects	
Windows	0.15	(0.96)	-	
UNIX	-0.04	(-0.26)	-	
All	0.02	(0.13)	-	
Linux	0.01	(0.05)	-	
Secret	-0.25***	(-2.92)	-0.20*	(1.67)
Published	0.51*	(1.72)	-0.17**	(1.90)
Location	-0.43	(-0.72)	-0.06	(0.67)
Time dummies (8)	Yes		Yes	
N	2772		2772	

*Notes: *** p < 0.01 * p < 0.10. +Estimates include security protection, confidentiality, integrity, availability, input_validation, boundary_condition, buffer_overflow, access_validation, exceptional_condition, config error, other_vuln. besides Windows, UNIX, all, Linux and others, which denote vulnerability specific effects.*

One of the limitations of the above method in understanding the effect of availability of patches and vulnerability information disclosure is that it is possible that both the effects could have been averaged over attacks that are irrelevant. For example there are many attacks that are not exploited by attackers because the vulnerabilities pertain to very old versions of the software. The descriptive statistics in table 3 reveal that vulnerabilities not exploited by attackers generally tend to be much older than those that were exploited. In the analysis that follows we specifically consider the effect of age of the vulnerability and age of the patch on the number of attacks on hosts per day. In order to account for the fact that the vulnerabilities that are exploited tend to be older on average, we use a second order polynomial of elapsed days from date of publication of vulnerability and elapsed days from release of patch to understand the effect of elapsed days from availability of patches (elapsed patch days) and elapsed days from availability of vulnerability information (elapsed publish days).

Time effect of patching and publishing:

The probability of a vulnerability being exploited is a function of the attacker's fixed cost to attack relative to the gains from attacking relative to the gains from attacking. We use the number of elapsed days from the date of publication of the vulnerability as a proxy for attacker's fixed cost to attack. With time, attackers gather more information about vulnerabilities and the resulting proliferation of tools to exploit vulnerability reduces the fixed cost to attack. Since the fixed costs decrease with time, more attackers exploit unpatched vulnerabilities over time.

The availability of patches decreases the proportion of hosts that are not patched. Increased patching makes it less probable that an attacker will try to exploit a vulnerability on a given host. We use the number of elapsed days from the release of the patch as a proxy for both these effects of patch availability. Since it is likely that newer versions of the vulnerable software are released with patches fixing vulnerabilities, elapsed days from patching could be a reasonable proxy for the proportion of hosts that are patched. For example, more recent build of Windows XP comes with service packs that contain patches for vulnerabilities. This would suggest that proportion of machines patched would steadily grow over time reducing the number of attacks. In order to study the time effects of patches and information disclosure we use the following specification:

We assume that the number of attacks that are observed on a host during a day, A_{it}, is normally distributed, but since the number attacks cannot be negative, we use the Tobit specification:

$$A_{it}^* = \alpha_i + \tau_t + \delta_1 nopatch_{it} + \delta_2 published_{it} + \delta_3 patched_{it} + \delta_4 (1 - nopatch_{it}) t_{patch}$$
$$+ \delta_5 [(1 - nopatch) t_{patch}]^2 + \delta_6 t_{pub} + \delta_7 t_{pub}^{\ 2} + u_{it} \equiv (\mathbf{X\delta} + u_{it}) \qquad (7)$$
$$A_{it} = \max(0, A_{it}^*)$$

In (7) above, u_{it} is the random disturbance with $u_{it} \sim N(0, \sigma^2)$. t_{patch} and t_{pub} are the elapsed patch months and elapsed publish months respectively, where these can take negative values and $t_{patch}^{\ 2}$ and $t_{pub}^{\ 2}$ are quadratic terms to capture non linearities in time effects.[12] We assign a common vulnerability fixed effect dummy to attacks that are not exploited in any time period. In case of vulnerabilities that were exploited, each has a vulnerability specific dummy. This implies that we have 59 vulnerability dummies α_i though there are 308 vulnerabilities in all. As in the case of the linear model, $published_{it} = 1$, in case of a *published* vulnerability (vulnerability that was published and not patched) and *secret = 1* in case of a *secret* vulnerability (vulnerability that was neither published nor patched). Time effects are included in the form of time dummies, τ_t. Table 6 provides the result of estimation.

Table 6 Tobit regression – Effect of elapsed patch days and elapsed publish months
Dependent variable – Average number of attacks per day per host (z-statistics in parentheses)

Variable	Coefficient Estimate
Not Patched dummy	-35.81*** (-4.72)
Patched & Published dummy	-34.32*** (-5.06)
Published & Not Patched dummy	19.39***(2.56)
Elapsed patch months**(1-Not Patched)*	0.39 (0.80)
Elapsed publish months	0.10 (0.25)
Elapsed publish months squared	-0.002 (-0.41)
Elapsed patch months squared**(1-Not Patched)*	-0.002 (-0.32)
Time dummy variable included (8)	Yes
Vulnerability dummy variable (59)	Yes
Vulnerability technical characteristics included	No
No. of observations	2772
Log likelihood	-632.20
No of vulnerabilities	308
σ (Std deviation)	12.69

*Notes: *** p < 0.01 ** p<0.05 * p < 0.10.*

Impact of elapsed patch and publish months

The results are explained using figures 1,2,3,4 and summarized using figure 5. These figures have been constructed using the predicted values of the tobit regression reported in table 6. In each of these figures

[12] Note that though all *secret* vulnerabilities are eventually disclosed (else they would not be in the sample), not all vulnerabilities are patched. Time elapsed since patch is thus not defined for unpatched vulnerabilities, and we handle this by only estimating the relevant coefficients for patched vulnerabilities, as shown in table 6.

X-axis represents time in elapsed calendar days and Y-axis represents $E(A_{it}) = \Phi(\frac{\hat{\mathbf{X}\boldsymbol{\delta}}}{\sigma})\mathbf{X}\hat{\boldsymbol{\delta}} + \sigma\phi(\frac{\hat{\mathbf{X}\boldsymbol{\delta}}}{\sigma})$, where $\Phi(.), \phi(.)$ and σ represent normal CDF, normal PDF and estimated variance respectively. Figure 1 shows the effect of publishing a vulnerability using a test case in which the vulnerability is published at *t=0*. In other words, the vulnerability depicted in the figure changed stats from *secret* to *published* at *t=0*. From figure 1, publishing a vulnerability sharply increases the expected number of attacks when published and then increases more gradually with time.

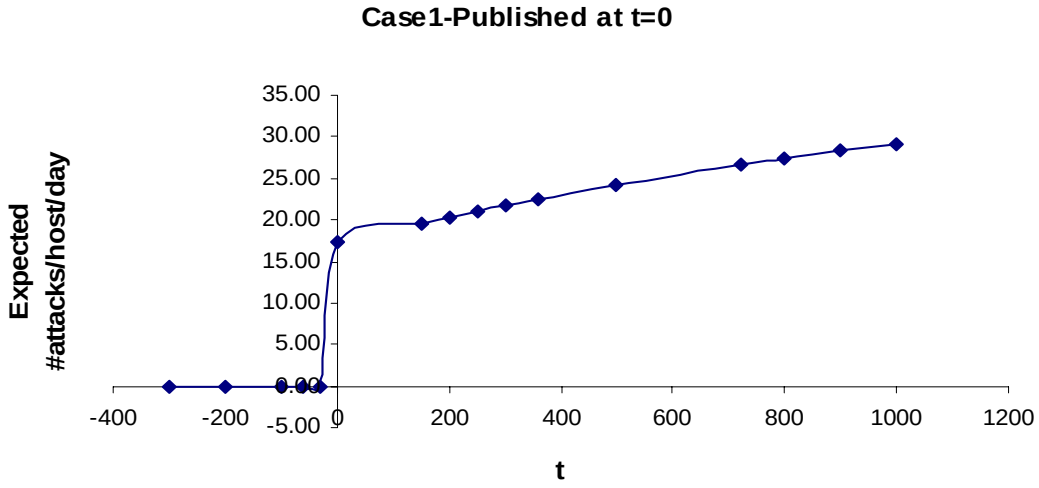

Figure 1 Simulated impact of publishing without patch

Figure 2 shows the impact of patching a vulnerability that was already *published* at *t= -300* but patched at *t=0*. The sharp drop in the expected number of attacks per day per host after patching of vulnerability clearly indicates that patching deters attackers. But after patching, with elapsed patch days the expected number of attacks per day per host increases with time, which may be an artifact of our specification.

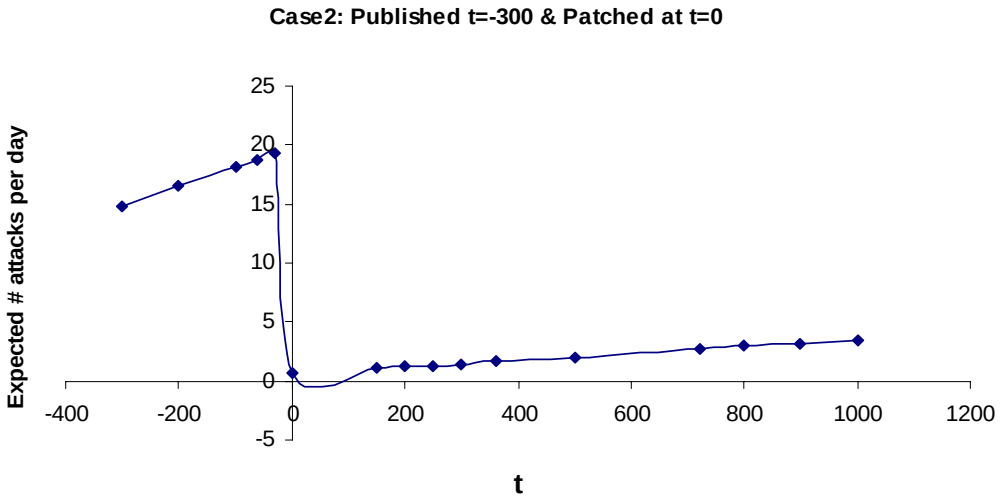

Figure 2 Simulated impact of patching a known vulnerability

The trajectory of secret vulnerabilities is depicted in figure 3. As expected *secret* vulnerabilities are attract the fewest attacks, though the closer the vulnerability gets to being published the expected number of attacks per day per host increases, albeit that the average number attacks is only around 0.02.

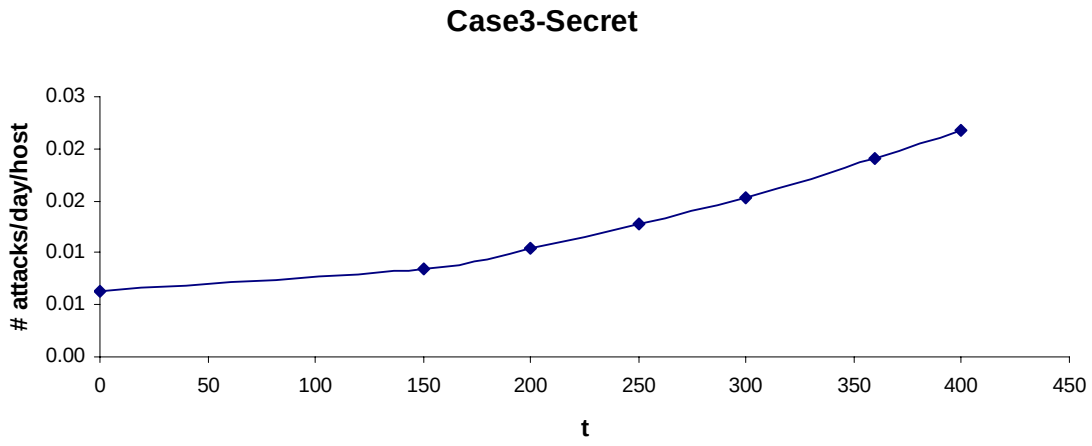

Figure 3 Simulated impact of secrecy

Vulnerabilities that were patched and published on the same day still get exploited more by attackers than *secret* vulnerabilities, as shown in figure 4. The figure depicts a small spike around at *t=0*. Further, the expected number of attacks per day per host increases with time, although the average number of attacks is still small, around 0.05

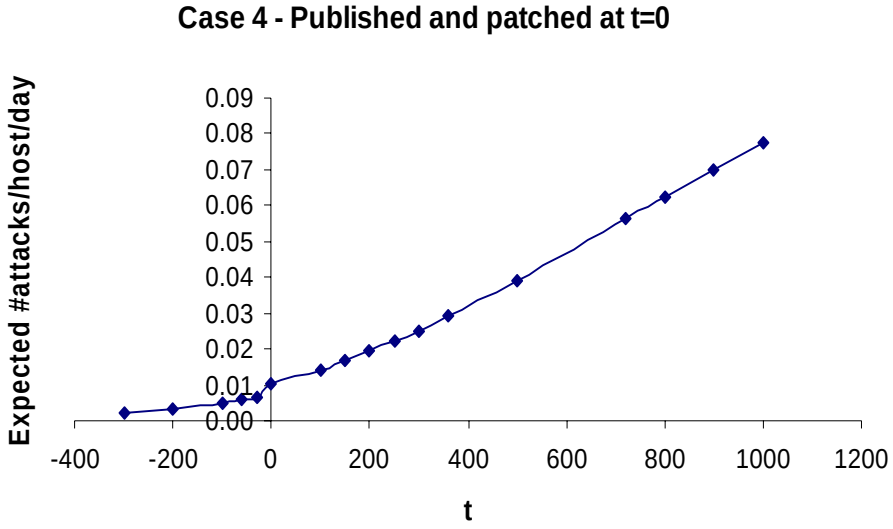

Figure 4 – Simulated effect of patching and publication

To summarize our results, both *patched* and *published* vulnerabilities attract more attacks than the *secret* vulnerabilities and *published* vulnerabilities that have no patches tend to get the most attacks. Further, with time, the number of attacks per day per host increases in the case of *patched* vulnerabilities but the number of attacks per day per host decreases with time in the case of *published* vulnerabilities. The results are summarized for a vulnerability published at *t=0* and patched at *t=200.*

Although our results indicate that secrecy is effective in reducing attacks, this should not be over-interpreted because our results do not address the question of damage, and neither do we entertain here the possibility of users mitigating or avoiding damage from attacks if informed of the vulnerability but without a patch. The results do underscore the importance of understanding user patching behavior. As noted, release of a patch appears to increase the number of attacks. This could be many users apparently

do not install the patch. It could also be that the patch helps attackers develop better exploits, but it remains true that unless attackers expect significant delays among a substantial fraction of users in installing the patch, there would be little point in attacking. Thus, a promising area for future research is to understand the factors that condition the speed with which users install patches, and in particular, the quality of the patch.

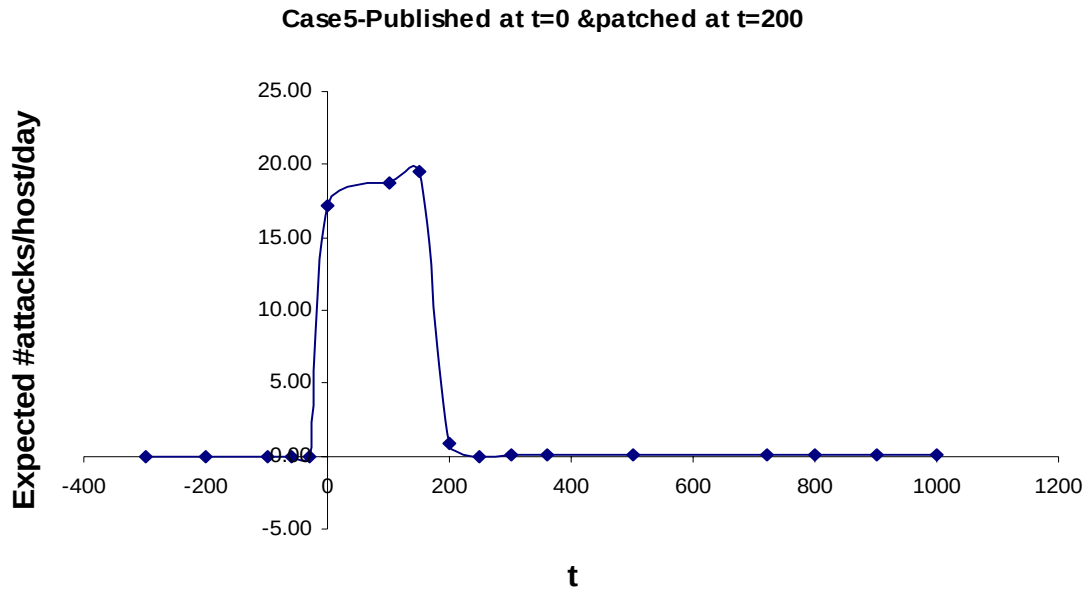

Figure 5- Simulated attack life cycle

3. Impact of vulnerability disclosure policy on vendors' patching behavior

This part of the paper presents an empirical analysis of the impact of various forms of vulnerability disclosure policies on when vendors release patches. We begin by discussing the data sources and then present the results.

3.1 Data

We acquired data from two sources. The first source is a sample of vulnerabilities published in SecurityFocus database. To generate a random sample of instantly published vulnerabilities and to get detailed vulnerability information, we used CVE ID from CVE ICAT database and matched it with a random sample of SecurityFocus online software vulnerabilities. SecurityFocus is one of the most popular software security online communities in the world. The vulnerability information published on SecurityFocus is mainly from the 'Bugtraq' mailing list, which serves as a forum for security experts to exchange up-to-the-minute information on viruses, security flaws and exploits. Currently, the SecurityFocus database contains more than 7000 vulnerabilities, mainly published from 1999 to 2004. 'Bugtraq' is a high volume, full disclosure mailing list for the detailed discussion and announcement of computer security vulnerabilities. Though the vulnerability information reported to Bugtraq is published instantaneously on Bugtraq, not all the vulnerabilities first published by Bugtraq correspond to instantaneous disclosure: Sometimes identifiers who discover the vulnerability notify vendors first and report to Bugtraq after a few days. In such cases, publication of vulnerability on Bugtraq is done only after the information about the patch fixing the vulnerability is acquired from vendors. In our data, we have tried to identify the instantly disclosed vulnerabilities by finding out the time when vendors are notified about the vulnerabilities (by verifying the date from other sources like visiting vendor site or

from CERT). The data so collated thus contains only vulnerabilities that would conform to the definition of *instantaneous* vulnerability disclosure.

The second source consists of data on vulnerabilities is acquired from CERT/CC.[13] As in the case with 'Bugtraq' vulnerabilities, we matched CERT/CC vulnerabilities with CVE numbers in the CVE ICAT database. CERT/CC gets the vulnerability reports from individuals, open source parties, vendors and research institutes. Vulnerabilities disclosed by CERT/CC fits the definition of *partial* disclosure as CERT/CC does not release all technical information about a vulnerability. CERT/CC typically notifies vendors first after getting the vulnerability reports and provides vendors 45 days to patch the vulnerability before making the vulnerabilities public (CERT/CC, 2000). Further, in almost all cases, CERT/CC discloses information about vulnerability only after vendors issue patches fixing the vulnerability.

We further augmented the data on vulnerabilities with statistical information about vendors such as firm type (Open Source or closed source), and sales and employment data from a variety of sources, including Yahoo! Finance, Hoover's Online and Dun & Bradstreet's Million Dollar Database. Thus each observation in the sample contains vendor characteristics and vulnerability characteristics, including vulnerability class, exploit range, loss type, exposed system type and exposed component type and severity[14].

Using the data gathered, the number of days taken by the vendor to patch the vulnerability has been calculated using the typical life cycle of a software product. A typical life cycle of a software product can be described as follows (Arbaugh et al., 2000): At time '0' the product is released by a vendor and used by users. A vulnerability is discovered by a benign user[15] and reported at t_0. Notice that if the vulnerability is reported to CERT/CC, then CERT/CC will contact vendor first, while if the vulnerability is reported to SecurityFocus, SecurityFocus will make it public immediately. Assume that vendors know about the vulnerability when they are notified about it or that they became aware on account of the vulnerability being published elsewhere. Disclosure policy T requires that this vulnerability is kept secret until time $t_0 + T$ and disclosed after that, so that $T = 0$ if the vulnerability is instantly disclosed. Vendors provide a patch[16] for this vulnerability at a calendar time $\tau + t_0$[17]. Attackers might find and exploit at time t_0+s. We take "notify vendor date" or "first known public date", which ever is earlier, as t_0. We use time τ to measure the time vendors take to patch the vulnerability.

[13] CERT/CC, a major reporting center for Internet security problems, is a federally funded research and development center (FFRDC) operated by Carnegie Mellon University.

[14] vulnerability natures include the following fields: lunch type (remote or local), vulnerability class (Input validation error, Access validation error, Exceptional condition handling error, Environmental error, Configuration error, Race condition, Design error and others), Loss type (loss of availability, loss of confidentiality, loss of integrity, loss of security protection and others), Exposed System Component (Operating system, Protocol stack, Server application, Non-server application, Hardware, Communication protocol, Encryption module and others) and Destination type (Windows system, Unix System, Apple System) and others.

[15] A benign user is one not interested in exploiting this vulnerability. Also note that if an attacker discovers the vulnerability first then it will immediately attack and any disclosure policy is a moot.

[16] We use a loose definition of "patching" to include not only the vendor's official patch, but also the release of a new version, workaround and any other type of remediation.

[17] Note that τ can be higher or lower than T if vulnerability is not disclosed immediately.

TABLE 7 Summary statistics by Year, Disclosure Policy, patched/un-patched, open/close source

	1999	2000	2001	2002	2003	Total
Avg. time to patch Measured by days	92.69 (139.40)	66.19 (142.09)	179.84 (603.65)	66.66 (164.13)	67.13 (226.02)	103.36 (365.77)
Patched/Total	62	39	61	43	27	41
	(CERT/CC) Partial disclosure policy			**(Buqtraq) Instant disclosure policy**		**Total**
Avg. days to patch	101.21 (373.36)			126.38 (271.91)		103.36 (365.77)
Patched/Total	40%			57%		41%
Employee Size (k)	43.07 (91.14)			40.46 (67.40)		42.92 (89.98)
Open source/Total	12%			11%		12%
Severity[18]	2.66 (0.58)			2.43 (0.62)		2.64 (0.59)
	UN-PATCHED			**PATCHED**		**Total**
Employee Size (k)	43.77 (95.66)			41.73 (81.33)		42.92 (89.98)
Open source/Total	11%			14%		12%
Severity	2.63 (0.61)			2.66 (0.55)		2.64 (0.59)
	Close source			**Open source**		**Total**
Avg. days to patch	449.98 (536.17)			376.52 (476.08)		441.21 (529.83)
Patched/Total	41%			47%		41%
Severity	2.65 (0.59)			2.61 (0.60)		2.64 (0.59)
Employee size (k)	46.48 (92.60)			3.89 (34.52)		42.92 (89.98)

Table 7 describes our sample. Table 7 suggests the following; (1) vulnerabilities in the CERT/CC and Bugtraq samples tend to be similar in that key variables like vendor size, vendor type and the severity level have similar average values in both samples. Moreover, a higher percentage of vulnerabilities are patched under instantaneous disclosure policy in our sample. (2), average vendor response time is longer for the vulnerabilities published on SecurityFocus than on CERT/CC, although the difference is not statistically significant. Moreover, as we shall discuss later, controlling for vulnerability specific factors, vendor response for Bugtraq reported vulnerabilities is quicker than for CERT/CC reported ones. (3) Although open source software vendors are on averaged $1/16^{th}$ the size of the average close source software vendors, we find that 5% more vulnerabilities are patched in the Open-source software than close source software. And among those patched, the average time to patch is about 73 days shorter for the open source vulnerabilities than the close source.

[18] There are three levels of severity defined in the CVE database. As discussed later, CERT/CC has a much finer gradation for vulnerabilities, with each vulnerability assigned a severity score between 0 and 180.

3.2 Model Specification and Estimation

The key question is how vendors respond to disclosure. Arora, Telang and Xu (2004) point out that the vendors incur two types of costs. One is the patch development cost. The more resources a vendor allocates to patching, the shorter is the time taken to patch. Obviously, patching cost is closely related to the nature of the vulnerability and the software. It is intuitive to suggest that the patching cost decreases with time. Because fewer resources need to be assigned to create the patch and hence costs are lower.

However, waiting also costs the vendors by exposing its customers to attack from time $t_0 + s$ to time $\tau + t_0$. Obviously, the expected cumulative customer loss over time from being attacked though the vulnerability is increasing with time. Currently, software vendors are not liable for information security breaches of their customers due to vulnerabilities in their products (Varian, 2000). But vendors still internalize some of the losses suffered by their customers in the form of reputation loss and loss of future sales, and increased maintenance or support cost. How much of the customer costs vendor internalizes depends on factors such as its size, the nature of the market, and the intensity and nature of competition, which we do not explicitly model. Instead, we assume that the vendor minimizes the following cost function by choosing τ, the patch release time (cf. Arora, Telang and Xu, 2004).

$$V = C(\tau) + \lambda\theta(\tau, T : X)$$

C is the cost of patching, τ is the time to patch, θ is the cost to the customer and λ is the fraction of loss hat vendor internalizes. T is the disclosure policy. Note that the disclosure policy T affects the customer cost. If vulnerability is disclosed without a patch then the customers are subject to attacks and hence their costs go up. This means that the vendor's cost increases as well (if $\lambda > 0$) and hence it will release the patch earlier (Arora, Telang and Xu, 2004). This is the main hypotheses we test with our data.

In our data, we have many un-patched vulnerabilities as well. Clearly, vendors make decision on whether to patch, and then when to patch it. If the cost of patching is too high compared to the loss to customer then vendor may never patch it. Therefore we conduct the empirical analysis with in two steps. First, we estimate whether vendors provide a patch and how this probability changes with disclosure policy and other key factors. Second, conditional on a patch, we estimate how the time to patch changes with disclosure policy.

3.3 Whether to patch

A vendor's decision of whether to patch is based on a comparison of patching cost and loss due to delaying patching. In short, if the optimal solution of the vendor cost function leads to $\tau^* = \infty$[19], then vendor never patches. We denote the vendor decision of whether to patch as PATCH =1 if the vendor decides to patch, and PATCH =0 if the vendor decides not to patch. Since we do not observe θ (t, X) or C or λ, we use a reduced form probit specification for the probability that vendor chooses to patch

$$P(PATCH = 1|t, X) = \beta_0 + \beta_1 inst\_dislosure + \beta_2 SALE + \sum_{k=1}^{m} \alpha_k Vul_k + u$$

where **inst_disclosure** is a dummy variable = 1if the vulnerability is published under instantaneous disclosure policy; **SALE** is the vendor's annual sales, a measure of size; $\mathbf{Vul}_k$ is a vector of the vulnerability characteristics; **u** is an *i.i.d.*, zero-mean normal disturbances. [20]

---

[19] In practice $\tau^* = \infty$ simply means that It should be large enough. As long as we not observe a patch within the time period of our data collection we treat it as un-patched vulnerability.

[20] We use the "CERT severity score" to measure the severity of vulnerability (It is part of $\mathbf{Vul}_k$ vector), which ranges between 0 and 180. However, we don't have this metric for vulnerabilities only published on Bugtrag. To generate the "CERT severity scores" for Bugtraq sample, we do the following. First we regress CERT severity scores (for the

The probit model estimate results are presented in Table 8. These results highlight the following:

i) The average probability of patching, controlling for the vendor types and vulnerability characteristics, is significantly higher under instantaneous disclosure policy than CERT/CC policy. Therefore, the results confirms that instant disclosure pushes the vendors to patch sooner. Note that the impact of inst_policy in a probit specification is

$$\Phi(\beta_0 + \beta_2 SALE + \sum_{k=1}^{m} \alpha_k Vul_k + 0.71) - \Phi(\beta_0 + \beta_2 SALE + \sum_{k=1}^{m} \alpha_k Vul_k)$$

where $\Phi(z)$ is the standard normal cumulative distribution function. In this sample, the average effect of instantaneous disclosure policy is estimated to increase the probability of patching by about 28.7%.

ii) Larger vendors are more likely to patch than small vendors.

iii) The vulnerability severity, measured by cert_severity_score, has a positive effect on PATCH, which suggests that sever vulnerabilities are more likely to be patched by the vendors.

iv) Vulnerabilities that can cause confidentiality loss are less likely to be patched. But vulnerabilities in non_server type of applications[21] are more likely to be patched.

Table 8 Probit model regression (Closed Source Sample Only)
Dependent Variable: PATCH

Variable	Coef.	(t-statistics)
Constant	-0.54	(-5.00)
Inst_disclosure	0.71**	(5.57)
SALE	0.003**	(6.73)
cert_severity_score	0.004*	(2.38)
launch_remotely	-0.13	(-1.47)
Confidentiality_loss	-0.28**	(-2.76)
integrity_loss	0.06	(0.60)
availabiliy_loss	-0.003	(-0.04)
operating_system	-0.013	(-0.22)
network_protocol_stack	0.42	(1.28)
server_application	0.002	(0.02)
non_server_application	0.29**	(3.12)
Hardware	0.65	(1.58)
Communication_protocol	-0.11	(-0.82)
encryption_module	-0.93	(-3.31)
Windows	0.02	(0.18)
Unix	0.08	(1.15)

Number of obs = 2608
Log likelihood =-1654.66; Pseudo R^2 = 0.0548

CERT sample of vulnerabilities) on various vulnerability characteristics. Then we use the estimated coefficients to predict the CERT severity score for vulnerabilities in the bugtraq sample, using those characteristics.

[21] Server applications are programs providing some service to other (client) programs (cf. http://dict.die.net/server/).

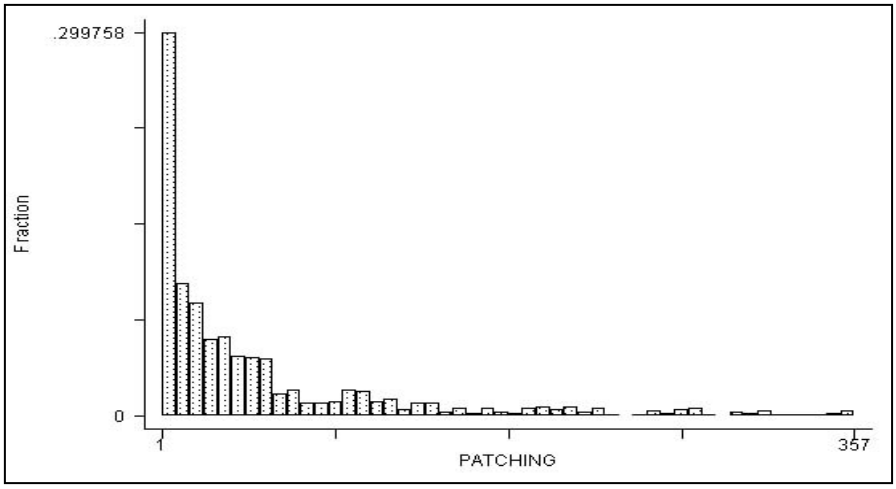

Figure 6 the distribution of τ (days)

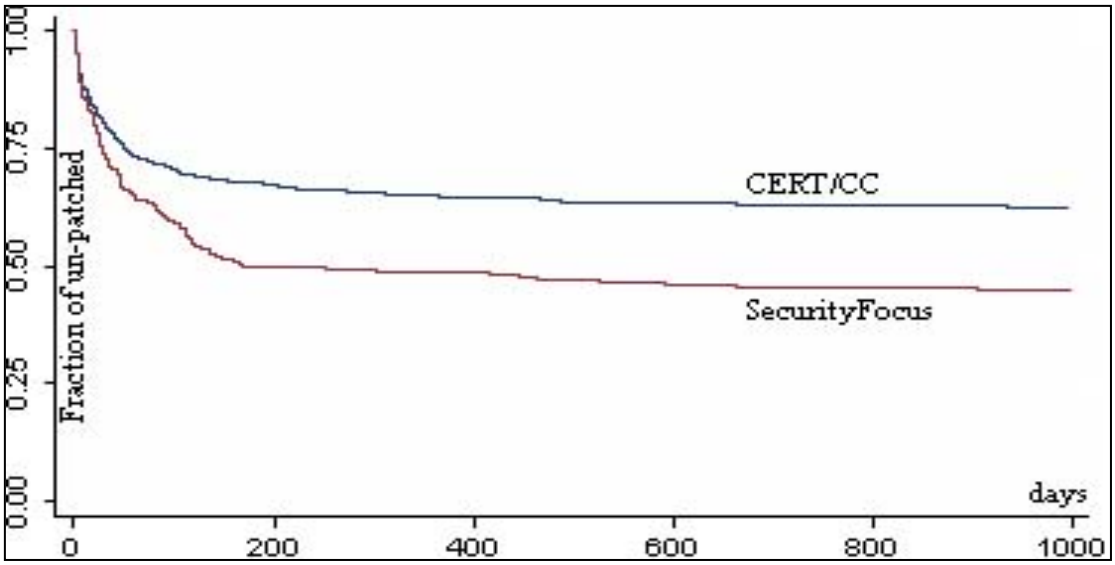

Figure 7: Proportion of unpatched vulnerabilities over time

When to patch

Now we test whether instantaneous disclosure policy plays a significant role in vendor's optimal patching time τ*(assume vendors do provide a patch). We first plot the patching time to help us understand the distribution of patching time across two policies. Figure 6 shows that the patching time in the sample resembles an exponential distribution. Figure 7 shows that there are significant differences in patching time between the vulnerabilities published on CERT/CC and Security Focus. Not only are a higher percentage of vulnerabilities patched under instantaneous disclosure policy, they are also patched faster.

Given the duration nature of our data, we estimate vendor's patching time using Proportional Hazards Model (Kiefer, 1988) with the patching time as the duration under risk. We use the Cox proportion model with a baseline hazard given by Box-Cox specification. The covariates (like disclosure policy, severity etc) proportionally shift the hazard function. We take the "time to patch" observation as duration, with each starts from some t_0 (normalized at 0) and ends at some $\tau + t_0$ (when the patch is available). Any un-patched vulnerability can be interpreted as right censored data. The estimation results with Proportional Hazards Model are presented in table 9. As in the previous part, we use the fitted Cert_severity_score to

represent the severity of vulnerability in estimation. Results from all observations are presented in column (1). Column (2) contains the estimation results from the close source sample only.

Table 9 Estimation of Vendor's patching behavior[22]
Dependent Variable = Time elapsed before patching

	All (1)		**Close source** (2)	
Variables	Haz. Ratio	Wald's test statistics	Haz. Ratio	Wald's test statistics
Inst_disclosure	1.69**	(5.18)	2.24**	(6.43)
Cert_severity_score	1.00	(1.37)	1.01**	(2.60)
Open_source	1.32**	(3.38)		
Launch_remotely	1.03	(0.27)	0.97	(-0.20)
Confidentiality_loss	0.61**	(-4.23)	0.68**	(-2.80)
Integrity_loss	0.85	(-1.30)	0.95	(-0.35)
Availabiliy_loss	0.84**	(-2.74)	0.90	(-1.41)
Operating_system	0.89	(-1.77)	1.01	(0.06)
Network_protocol_stack	1.65	(1.36)	2.09	(1.86)
Server_application	1.06	(0.48)	1.14	(0.97)
Non_server_application	1.52**	(3.82)	1.80**	(4.27)
Hardware	1.05	(0.11)	1.34	(0.56)
Communication_protocol	0.98	(-0.15)	1.02	(0.10)
Encryption_module	0.16**	(-4.22)	0.21**	(-3.30)
Windows	0.93	(-0.70)	0.91	(-0.80)
Unix	1.04	(0.51)	1.05	(0.57)
Sales			1.00**	(5.08)

* significant at 5% level. ** significant at 1% level.

The hazard ratio is simply exponent of estimated coefficients. If it is greater than 1 then that parameter decreases the "time to patch" and hence leads to a quicker patch. Instantaneous disclosure policy in both columns is positive and significant. So, in our sample, vulnerabilities published on Bugtraq not only are patched with higher probability (see the probit results in table 8), they are also patched significantly faster. Open source is estimated to be positive and significant suggesting that open source vendors respond faster than close source vendors. The focus of our analysis is not necessarily to argue about the relative efficacy of open source software products. However, our results clearly suggests that open source vendors are more responsive in patching disclosed vulnerabilities. Also note that the hazard ratio of instantaneous disclosure policy in column (2) is greater than in column (1), which suggests that the instantaneous disclosure policy has larger impact on the close source vendors than on open source vendors.

The vulnerability severity in column (2) is positive and significant. Therefore, vendors' patching time for more critical vulnerabilities is shorter. As expected, vendors patch the severe vulnerabilities quickly. Vendor size (measured by sale in the close source group) is positive and significant. Therefore, large vendors patch faster then smaller ones. Clearly, large established vendors have more to lose in terms of reputation and hence respond more effectively. Insofar as size is correlated with market power, the results indicate that the lower costs associated with size dominate any disincentive effect due to market power. Alternatively, larger firms may patch more quickly because the loss is proportional to the number of users whereas the cost of patching is not.

[22] Proportional hazard model is applied in estimation with "time to patch" as the dependent variables. The other variables are defined same as table 8

Some vulnerability characteristics are also estimated to be significant in both models. Vulnerabilities that the loss type is "confidentiality" or "availability" are expected to be patched slower. And vulnerabilities in server applications are expected to be patched faster. Probably, vulnerabilities in the server applications cause higher losses prompting vendors to patch earlier.

The above analysis was done with all vulnerability data (patched or unpatched). We also repeat the analysis with only the patched vulnerabilities. The policy variable is then significant at 10% confidence interval and the sign and general magnitude remains the same.

To summarize, we find that first, instantaneous disclosure policy makes vendor's more likely to patch their product. Therefore, the instant disclosure parties, like SecurityFocus, may contribute in an important way to the problem of managing software security[23]. Second, the optimal patching time of the vendors is smaller in instantaneous disclosure policy than partial disclosure policy, which means vendors patches more quickly under instantaneous disclosure policy. Third, open source vendor are more likely to be patch than close source. Finally, average optimal patching time for open source software is shorter than close source software.

4 Summary & Conclusion

Our paper provides critical empirical estimates on attacker and vendor response. To our knowledge, this is the first systematic empirical examination of either of these questions, and marks a contribution towards understanding how vulnerability information should be disclosed. Our results indicate that early disclosure forces vendors to patch earlier. At the same time, the first set of results imply that disclosing a vulnerability increases the frequency of attacks. Clearly, there is a tradeoff in disclosing vulnerability. The release of a patch for a known vulnerability decreases the number of attacks but the release of a patch for a hitherto unkown vulnerability increases the number of attacks. Thus, even when a patch is available, disclosing a vulnerability increases the frequency of attacks. Clearly, attackers believe that not all users will patch in time. One must not rush to draw conclusions about public policy since, as noted, we lack information on successful attacks (e.g., those that would overcome the countermeasures that may exist in reality). We also lack information on the severity of damages. Finally, our results exclude by construction the possibility of users responding to vulnerability disclosure (without patches) in ways that would mitigate or avoid loss from attacks.

Our second set of results provide insights into how vendors respond. We find that open source vendors are quicker to patch when vulnerability is found in their products. Given the attention on the open source vs. close source debate, this is an important finding since the quality of a software product also depends on the ex-post support a vendor provides (Arora, Caulkins and Telang 2003. Similarly, large vendors are more responsive to defects in their products. Vendors are also more responsive to more severe vulnerabilities. Our results are therefore broadly consistent with a rational model in which vendors internalize some, but not all, of the customers' losses.

While our results are interesting, there are a number of qualifications. First, we only observe attack frequency and not the actual loss. While higher frequency would correlate with more losses, a more precise analysis with loss information would be an interesting future work. Second, Honeypots data could be biased. Trivial attacks may be over-represented, and more sophisticated attacks or internal attacks launched from within an organization are under-represented. Similarly, there may be important unobserved differences across vulnerabilities reported by Bugtraq and CERT/CC that may confound our findings on instant disclosure. Given the importance of these issues and little empirical work, we hope that our study paves the way for more research with new and possibly better data sources.

[23] We reiterate the fact that such instant disclosure need not be socially optimal (cf. Arora et al, 2004).

REFERENCES

Arbaugh, W. A., W. L. Fithen, and J. McHugh, “Windows of vulnerability: A case study analysis,” IEEE Computer, vol. 33 (December 2000), pp. 52–59

Arora Ashish, Jonathan P. Caulkins, Rahul Telang, “Sell First Fix Later: Impact of Patching on Software Quality,” Carnegie Mellon University, working paper, Jan. 2003

Arora Ashish, Rahul Telang and Hao Xu, “Timing Disclosure of Software Vulnerability for Optimal Social Welfare,” Carnegie Mellon University working paper, April 2004

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

Farrow, R., The Pros and Cons of Posting Vulnerability, The Network Magazine, www.networkmagazine.com/shared/article, 2000

Gordon. S and Richard Ford, “When Worlds Collide: Information Sharing for the Security and Anti-virus Communities,” IBM research paper, 1999

Varian H R, “Managing Online Security Risks,” The New York Times, http://www.nyt imes.com/library/financial/columns/060100econ-scene.html, 2000

Hoo, K. S., “How Much Is Enough? A Risk Management Approach to Computer Security,” Workshop on Economics and Information Security, University of California, Berkeley, CA, 2000

Spitzner L. “Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community”, Addison-Wesley. Honeynet Project, 2000

Lawrence A. Gordon, Martin P. Loeb and Tashfeen Sohail, "A framework for using insurance for Cyber-Risk Management" Communications of the ACM Volume 46(March, 2003), pp. 81 - 85

Nicholas M. Kiefer, “Economic Duration Data and Hazard Functions” Journal of Economic Literature, Vol.26, No.2 (Jun., 1988), pp. 646-679

Schneier, B., Computer Security: It's the Economics, Stupid. Workshop on Economics and Information Security, University of California, Berkeley, CA, 2002

Steve Beattie, S. A., Crispin Cowan, Perry Wagle, and Chris Wright, Timing the Application of Security Patches for Optimal Uptime. Proceedings of LISA 2002: 16th Systems Administration Conference, Philadelphia, PA, 2002

Stuart E. Schechter and Micheal D. Smith, "How Much Security is Enough to Stop a Thief? The Economics of Outsider Theft via Computer Systems and Networks." Harvard working paper, Harvard University, MA, 2000