An Empirical Analysis of Software Vendors' Patching Behavior: Impact of Vulnerability Disclosure[1]

Ashish Arora, Ramayya Krishnan, Rahul Telang, Yubao Yang

{ashish, rk2x, rtelang, yubaoy}@andrew.cmu.edu

H. John Heinz III School of Public Policy and Management

Carnegie Mellon University, Pittsburgh PA 15213

This version: January, 2006

Abstract

One key aspect of better and more secure software is timely and reliable patching of vulnerabilities by software vendors. Recently, software vulnerability disclosure, which refers to the publication of vulnerability information before a patch to fix the vulnerability has been issued by the software vendor, has generated intense interest and debate. In particular, there have been arguments made both in opposition to and in favor of alternatives such as full and instant disclosure and limited or no disclosure. An important consideration in this debate is the behavior of the software vendor. How quickly do vendors patch the vulnerabilities in general and after disclosure in particular? This paper compiles a unique data set from CERT/CC and SecurityFocus to answer this question. Our results suggest that disclosure policy has a significant positive impact on the vendor patching speed. Vendors are 137% more likely to patch due to disclosure. In particular, *instant disclosure* hastens the patch delivery by almost 29 days. Open source vendors patch more quickly than closed source vendors and severe vulnerabilities are patched faster. We also find that vendors respond more slowly to vulnerabilities not handled by CERT/CC. This might reflect unmeasured differences in the severity and importance of vulnerabilities. It might also reflect the stronger lines of communication between CERT/CC and vendors, and the value of the vulnerability analysis by CERT/CC.

***Keywords**:* Security vulnerability, disclosure policy, patching speed, open source, hazard functions

[1] Authors would like to thank the staff of CERT/CC for the data as well as valuable insight into disclosure process. We also thank participants of WEIS'05 (Workshop on Economics of Information Security) held in Boston, MA and WISE'05 (Workshop on Information Systems and Economics) held in Irvine, CA for their useful feedback.

1. Introduction

Information security breaches pose a significant and increasing threat to national security and economic well-being. In the Symantec Internet Security Threat Report (2003), each company surveyed experienced an average of about 30 attacks per week. Anecdotal evidence suggests that losses from such cyber-attacks can run into millions of dollars.[2]

These attacks often exploit software defects or vulnerabilities. Over the last few years, the number of vulnerabilities found and disclosed has exploded. 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 3,782 reports of vulnerabilities in the year 2003 alone and has reported more than 82,000 incidents involving various cyber attacks. Software vendors, including Microsoft, have announced their intention to increase the quality of their products and reduce vulnerabilities. Despite this, it is likely that vulnerabilities will continue to be discovered and disclosed in foreseeable future.

One key aspect of better and more secure software is timely and reliable patching of vulnerabilities by vendors. Patching can be viewed as ex-post product support and an important part of software product life cycle. However, while vendors' incentives to release timely, high quality software has been well studied (Krishnan et al 2000), patching is under-appreciated and under investigated component of overall software quality and security (Arora, Caulkins, Telang 2005). The issue of patching vulnerabilities has gained even more prominence because of public disclosure of these vulnerabilities. Typically, once a vulnerability is found by a third party and made known to the vendor, the vendor is expected to release a timely patch and disclose the vulnerability information along with the patch. However, the process of vulnerability disclosure is controversial. Many users feel that the vendors are not responsive enough and patches are of poor quality and information accompanying them is inadequate. Many of these users prefer to disclose vulnerability information upon discovery and before the patch is out. The argument is that the threat of quick disclosure increases public awareness, makes public the information needed for users to protect themselves, puts pressure on the vendors to issue patches quickly, and,

[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.

over time, results in better quality software. This belief fuelled the creation of full-disclosure mailing lists, such as *Bugtraq* by SecurityFocus, in late 90's. But many believe that the disclosure of vulnerabilities, especially without a good patch, is dangerous for it leaves users defenseless against attackers.[3]. Richard Clarke, President Bush's former special advisor for cyberspace security, criticizing full disclosure said: "It is irresponsible and sometimes extremely damaging to release information before the patch is out."[4]

Vulnerability disclosure is a key public policy issue. CERT/CC, an influential policy making organization and probably the most important conduit for disclosing vulnerabilities, favors a cautious approach to vulnerability disclosure. After learning of a vulnerability, CERT/CC contacts the vendor(s) and provides a time window, which we refer to as a disclosure window, within which to patch the vulnerability. The declared policy (and the de facto policy before October 2000) is to give vendors a 45 day disclosure window. Typically, the vulnerability is disclosed when the window closes or a patch is made available, whichever comes first.

Other organizations have proposed their own policies. For example, OIS (Organization for Internet Safety), which represents the consortium of 11 software vendors has suggested its own guidelines (mainly that the vendors be given 30 days).[5] In addition, firms such as iDefense and TippingPoint/3Com buy vulnerability information from users for their clients and have their own policies for disclosing vulnerabilities.

While there is good theoretical understanding of how disclosure affects software vendors' patching behavior (see Arora, Telang and Xu 2004 for details), there is little empirical understanding of vendors' patching behavior in general and impact of disclosure in particular. Thus the major goal of our paper is to empirically estimate (i) the key factors that affect vendors' patching decision and (ii) whether (and by how much) disclosure induces vendors to patch faster. A sensible disclosure policy is possible only if we can quantify vendors' response to disclosure.

[3] One of the recent examples is the Cisco IOS (Internetwork Operating System, the OS that runs Cisco's routers and some of its switches) vulnerability. The exploitation techniques of this vulnerability were disclosed at Black Hat conference in July 2005 by an individual researcher, provoking a law suit from Cisco against the researcher and the conference organizers, and an injunction to prevent further public discussion. .Customers were warned by Symantec that information revealed at the conference "represents significant threat against existing infrastructure currently deployed". See http://www.informationweek.com/story/showArticle.jhtml?articleID=166403842
[4] See http://www.blackhat.com/html/bh-usa-02/bh-usa-02-speakers.html#Richard Clarke.
5 Members include @stake, BindView, Caldera International (The SCO Group), Foundstone, Guardent, ISS, Microsoft, NAI, Oracle, SGI, and Symantec. For more details see http://www.oisafty.org

To do so, we compile a unique dataset of nearly 450 vulnerabilities, published by National Vulnerability Database (previously the NIST ICAT metabase) from September 26, 2000 to August 11, 2003. Since a vulnerability can involve more than one vendor, our empirical analysis is based on 1469 observations, each involving a vulnerability-vendor pair. For each vulnerability-vendor pair, we gather information on when the vendor was notified, if and when a patch was made available,[6] if and when the vulnerability was publicly disclosed, the nature and severity of the vulnerability and characteristics of the vendor such as size.

We find that *disclosure* increases the vendor's patching speed considerably. A vendor is 137% more likely to patch after disclosure than before. In particular, if a vulnerability is disclosed instantly then, on an average, the patch comes 29 days earlier than without disclosure. We find that *open source vendors* patch more quickly than closed source vendors and *severe* vulnerabilities are patched faster. More interestingly, we find that vendors respond more expeditiously to vulnerabilities that are processed and published by CERT/CC. This might reflect unmeasured differences in the severity and importance of vulnerabilities. It might also reflect the stronger lines of communication between CERT/CC and vendors, and the value of the vulnerability analysis by CERT/CC.

The rest of the paper is organized as follows. Section 2 reviews the related literature. Section 3 sets out the economic model of the vendors' patching decision; Section 4 provides a description of the data and variables; Section 5 presents the empirical model and the estimation results with discussion. We conclude and discuss limitations of this research in section 6.

2. Literature review

Our work draws from the literature on software quality and how process improvements lead to a higher quality. Krishnan et al (2000) show that having better personnel, more use of case tools, and more up front investments in planning, design, etc. improve product quality. However, patching is not widely studied. Baker et al (1998) focuses on the ex-post software support, namely software maintenance and examine how complexity and programmer capabilities affect maintenance costs. Again, the focus is not on patching per se.

[6] A software patch can be an upgrade (adding increased features), a bug fix, and a new hardware driver or update to address new issues such as security or stability problems. Most patches are free to download. But in come cases, the customers need to purchase the newer version at a discounted upgraded price (and requiring a reinstallation of the program) to have the vulnerability fixed. (www.softwarepatch.com)

The second stream of work we draw from is from the literature on economics of information security. Anderson (2001) provides a good overview of the issues related to economics of information security. The key distinction is that most of the work (which is still in the nascent stage) in the area of economics of information security is analytical and theoretical in nature while our paper brings data to some of these theories, thus supports better decision making by policy makers and managers.

Gordon and Loeb (2002) propose a framework for optimal investment in information security. Chen, Kataria and Krishnan (2005) suggested a software diversification based strategy for increasing the system security via reduction in system downtime. Several papers in the literature examine the effects of creating a market for vulnerabilities. Camp and Wolfram (2004) describe how a market for vulnerabilities may be created in order to increase the security of systems. Kannan and Telang (2004) employ a formal model to examine whether a market based mechanism is better than the setting in which a public agency (CERT/CC) acts as an intermediary. They show that markets always perform worse than CERT/CC because privately optimal disclosure rules are socially sub-optimal. Ozment (2004) shows, along the same lines, how such a market can function as an auction.

Our paper is more directly related to Arora, Telang, and Xu (2004), which develops a formal model using general functional forms to analyze vulnerability disclosure policy. Cavusoglu et. al. (2004) develop a similar model using specific functional forms. Choi, Fershtman and Gandal (2005) examine how software vulnerabilities affect firms that sell software and consumers that purchase software. In particular, they model a firm's decision on upfront investment in the quality of the software to reduce potential vulnerabilities, whether to announce vulnerabilities, and whether to apply a patch. A rational consumer decides on whether to purchase the software. Nizovtsev and Thursby (2005) model the incentives to disclose software vulnerabilities through an open public forum. Telang and Wattal (2004) show, using an event study, that disclosure of vulnerability information lowers the stock prices of software vendors, especially if a patch is not released at the same time, suggesting that disclosure is costly to vendors.

The model developed by Arora, Telang and Xu (2004) identifies a number of factors which condition whether early disclosure is optimal, including whether such disclosure allows users to protect themselves through workarounds and the probability of the vulnerability being rediscovered by a black-hat (Rescorla, 2004). A key insight from the model is that a *necessary* condition for disclosure to be effective is that it should prompt vendors to patch earlier. In this paper we measure the extent to which disclosure hastens the arrival of a patch. Systematic empirical testing has been hampered by lack of reliable data (Gordon et al., 1999). We fill this gap in the literature by developing a data set using data from CERT/CC and SecurityFocus and combining it with publicly available data on vendors.

3. Vendor's Patching Decision

To understand vendor's patching decision, we need to consider the costs vendor faces when considering patching decision. The vendor trades-off two major costs. First, there is the direct cost of developing a patch. To develop a patch, vendor commits resources and all else equal, the faster a patch is released, the more it costs. Thus if time to release a patch is τ then cost of patching is $C(\tau)$, where $C(.)$ is decreasing in τ. Second, the vendor's customers suffer losses if that vulnerability is exploited by attackers. Thus when a vulnerability is found and becomes public, hackers can target vendors' customers and they are more likely to suffer if the patch is not ready. Clearly, the longer a vendor delays the patch, the more likely the vulnerability will be discovered by attackers and the higher the customer loss. Let the customer loss function be $L(\tau; X)$ where X are other exogenous factors that condition customer loss, such as the severity of vulnerability and market size. However, vendor probably does not care about all the customer losses. It internalizes a fraction, λ, of customer losses due to loss of reputation, loss of future sales, contractual service obligations or legal liability. Thus, a vendor chooses patching time to minimize

$$V = C(\tau) + \lambda\, L(\tau,X)$$

The internalization parameter λ, depends on many factors including how competitive the market it. Presumably in a more competitive market, we would expect vendor to react faster.

We conjecture that the factors that increase patching costs should increase the patching time and the factors that increase customer losses should decrease the patching time. In the following we identify some key factors that affect vendors patching time. Note that we do not actually observe C and L and hence our model should be interpreted as reduced form in nature.

Disclosure

Disclosure increases customer loss $L(\tau,X)$ because disclosure makes it easier for the hackers to find the vulnerabilities (see Arbaugh et al 2000; Arora et al 2004 for details). While even a secret vulnerability may be discovered and exploited by black-hats, the likelihood of such exploitation is surely higher for a publicly known vulnerability. That implies that users face larger expected losses per unit of time that the patch is delayed if the vulnerability is public than if it is secret. Since vendors care about the losses suffered by users via λ, this implies that the marginal benefit of reducing the time taken to release a patch is higher when the vulnerability has been disclosed. Thus, disclosure should reduce expected patching time. Thus we hypothesize that

H1: Disclosure reduces vendor's patching time.

However, for actionable policy recommendation, we also need empirical estimates of how disclosure affects vendors patching time. Once this estimate is quantified, a policy maker like CERT/CC or OIS can calibrate and refine its policies.

Vendor Size:

Large vendors typically have higher sales volume and large customer base (not directly controlled for in our analysis), which implies higher customer loss from the disclosure of a vulnerability. Large vendors may also care more about reputation loss because of the spillover effect on other products sold by the vendor. Hence larger vendors may have higher λ. Large vendors may also patch faster if they are better able to afford larger patch development teams. Since patching costs do not vary significantly with the number of software users, the model suggests that large vendors should patch faster (both higher L and higher λ, and perhaps smaller marginal cost of patching). Thus we hypothesize

H2: Larger vendors patch faster.

Severity Level of Vulnerability:

CERT/CC measures the severity of the software vulnerability using a severity score called *severity metric,* which ranges from 0 and 180, that assigns an approximate severity to the vulnerability. This score reflects several factors including the scope, difficulty of exploit and impact of the vulnerability[7]. Since the extent of loss is likely higher with more severe vulnerability, it follows that more severe vulnerabilities should be patched faster.

H3: Severe vulnerabilities are patched faster.

Public Firms:

We expect that publicly traded firms should patch faster. Managers in public firms may experience the consequences of a fall in reputation more directly through reductions in share price (see Telang and Wattal 2004). Thus publicly traded firms may perceive a higher benefit from patching faster.

H4: Publicly traded firms patch faster.

CERT Effect:

While the key focus of our paper is on how disclosure affects vendor response, it is likely that vulnerabilities handled by CERT/CC are responded to differently. An organization like CERT/CC has higher reputation, and its pronouncements have more visibility. Moreover, it does its own investigation before contacting vendors. Thus, vendors may respond more favorably to CERT/CC than to SecurityFocus or other sources. Moreover, there might be selection issues in that CERT/CC may deal with more severe vulnerability while SecurityFocus and others are less discriminating in their disclosures.[8] Despite controlling for severity, there may still be some unobserved variations in the degree of "severity" across vulnerabilities. This may lead to spurious estimate on disclosure. Therefore, we include CERT dummy as a control variable and hypothesize that

H5: Vulnerabilities handled (i.e. eventually published) by CERT/CC are patched faster.

Source of disclosure:

Besides CERT effect, we also investigate the effect of the *source of disclosure*. For example, even though the vulnerability is eventually published by CERT/CC or SecurityFocus, it may be disclosed publicly by some other party before its eventual publication (for example, a vendor may disclose the vulnerability before other vendors have finished patching). Similarly, not only

[7] Detailed information on the CERT/CC Metric can be found at http://www.kb.cert.org/vuls/html/fieldhelp
[8] In fact, from our data we know that the CERT/CC, because it does its own research on severity of vulnerability, concerns itself with somewhat more severe vulnerabilities only.

may CERT/CC publish the vulnerability, it may also be the first to disclose it. Thus, in this paper, we make distinction between where the vulnerability are eventually *published* and who first *discloses* the information. The CERT effect tests for whether vulnerability eventually handled by CERT/CC are patched faster. The *source of disclosure* effect controls for the heterogeneity in *disclosure source*. Similar to the argument made in the CERT effect, we expect that

H6: Vulnerabilities (first) disclosed through CERT/CC publication are patched faster than vulnerabilities published by CERT/CC but first disclosed by other sources.

Open Source:

There is a stream of work arguing that open source vendors provide better quality and are more responsive to customers, while others have argued the contrary.[9] However, we are not aware of systematic empirical work that estimates the difference in open source and closed source vendors in patching speed. Thus, we also investigate the nature and extent of the difference between open source vendors and others.

4. Data and variables

4.1 Data sources

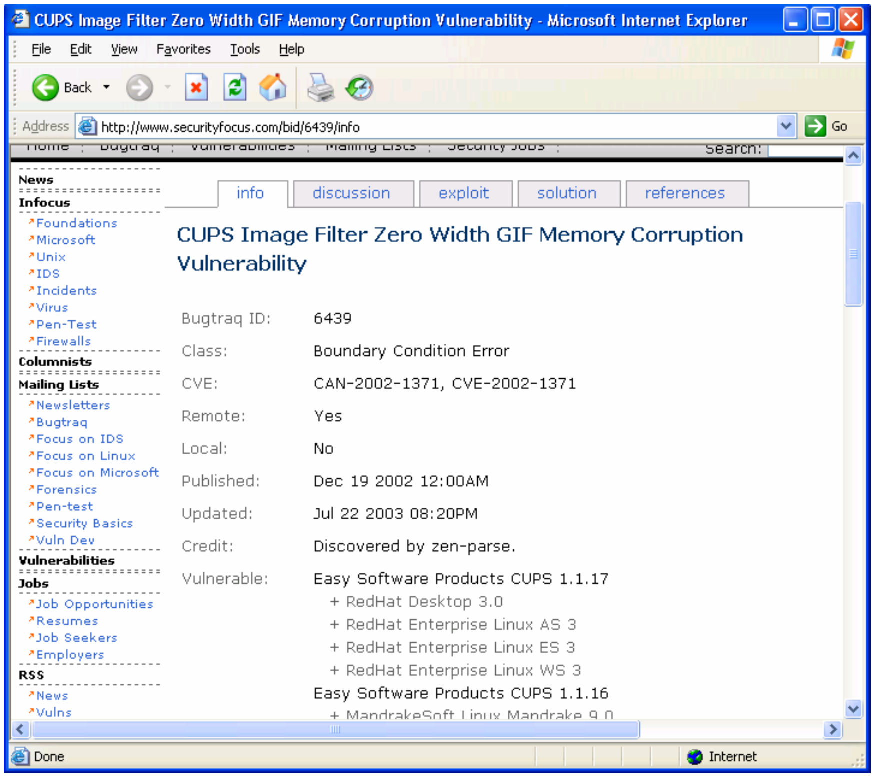
The vulnerabilities studied in our research are from the two most important vulnerability disclosure sources, CERT/CC and SecurityFocus. For each vulnerability-vendor pair, we collected information about whether the vendors released a patch and how much time they took to do so. We further augmented the data by adding the information about vendor type and vendor size, from the vendor's website or the Hoover's online database. We also added standard technical information about each vulnerability which we acquired from the National Vulnerability Database (NVD). NVD is a product of the Computer Security Division at the National Institute of Standards and Technology (NIST). NIST assigns unique identifiers known as CVE ID to vulnerabilities that are known publicly and records the standard technical information pertaining to the vulnerability as well as the related software.

CERT/CC and SecurityFocus are the two most important information sources of the National Vulnerability Database. About 60% of the vulnerabilities in NVD are published by

[9] See http://www.personal.psu.edu/users/r/p/rpp124/openandclosed.htm

SecurityFocus and about 10% are by CERT/CC. Between 9/26/2000 to 8/11/2003, CERT/CC published 526 vulnerabilities notes, involving 622 vendors. From these 526 vulnerabilities, we selected the 465 vulnerabilities that are documented by NVD. Note that the same vulnerability can be published by both CERT/CC and SecurityFocus, albeit at different dates. From SecurityFocus we randomly pick 131 NVD documented vulnerabilities that are *not* included in CERT/CC vulnerability notes. Figure 1 and 2 are examples of the vulnerability publication by SecurityFocus and CERT/CC.

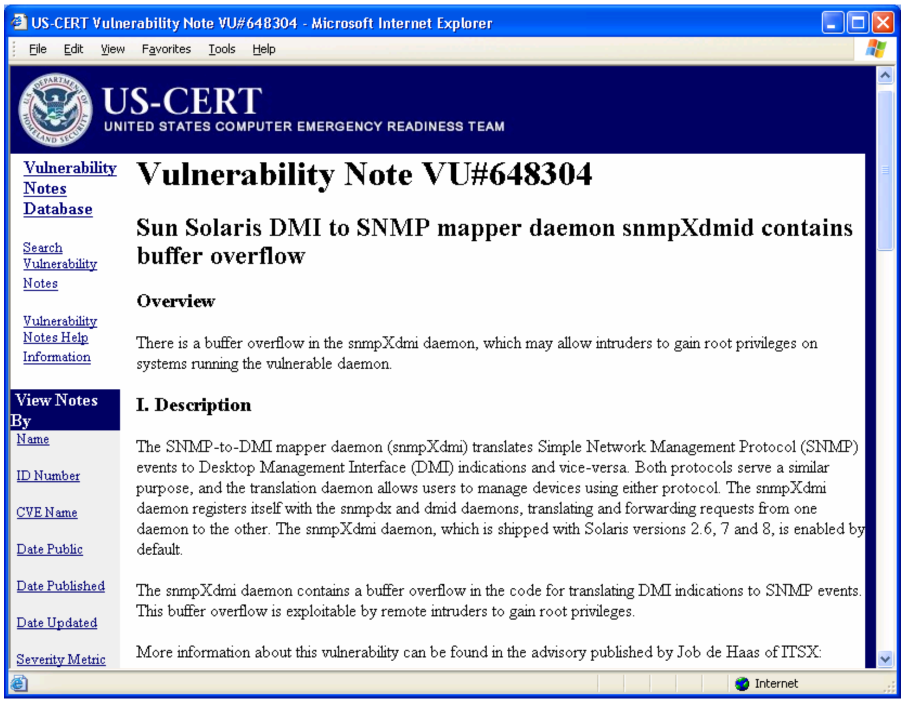
Figure 1: Example of vulnerability publication by SecurityFocus



CUPS Image Filter Zero Width GIF Memory Corruption Vulnerability

Bugtraq ID:	6439
Class:	Boundary Condition Error
CVE:	CAN-2002-1371, CVE-2002-1371
Remote:	Yes
Local:	No
Published:	Dec 19 2002 12:00AM
Updated:	Jul 22 2003 08:20PM
Credit:	Discovered by zen-parse.
Vulnerable:	Easy Software Products CUPS 1.1.17 + RedHat Desktop 3.0 + RedHat Enterprise Linux AS 3 + RedHat Enterprise Linux ES 3 + RedHat Enterprise Linux WS 3 Easy Software Products CUPS 1.1.16 + MandrakeSoft Linux Mandrake 9.0

Figure 2: Example of vulnerability publication by CERT/CC


US-CERT Vulnerability Note VU#648304 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM
Vulnerability Notes Database
Search Vulnerability Notes
Vulnerability Notes Help Information
View Notes By
Name
ID Number
CVE Name
Date Public
Date Published
Date Updated
Severity Metric
Internet

Vulnerability Note VU#648304

Sun Solaris DMI to SNMP mapper daemon snmpXdmid contains buffer overflow

Overview

There is a buffer overflow in the snmpXdmi daemon, which may allow intruders to gain root privileges on systems running the vulnerable daemon.

I. Description

The SNMP-to-DMI mapper daemon (snmpXdmi) translates Simple Network Management Protocol (SNMP) events to Desktop Management Interface (DMI) indications and vice-versa. Both protocols serve a similar purpose, and the translation daemon allows users to manage devices using either protocol. The snmpXdmi daemon registers itself with the snmpdx and dmid daemons, translating and forwarding requests from one daemon to the other. The snmpXdmi daemon, which is shipped with Solaris versions 2.6, 7 and 8, is enabled by default.

The snmpXdmi daemon contains a buffer overflow in the code for translating DMI indications to SNMP events. This buffer overflow is exploitable by remote intruders to gain root privileges.

More information about this vulnerability can be found in the advisory published by Job de Haas of ITSX:

After excluding observations for which reliable patching time could not be established or for which it could not be determined whether the vendor is vulnerable[10], or when a vendor was the first one to report the vulnerability (along with a patch) to SecurityFocus or CERT/CC, our final sample contained 1469 observations, related to 325 vendors and 438 unique vulnerabilities. Note that a vendor may be exposed to multiple vulnerabilities and a vulnerability may affect multiple vendors. Some vulnerabilities are discovered and disclosed by vendors themselves, typically by releasing a patch. We excluded all such vulnerability vendor pairs from our dataset, because if a vendor is the first one to disclose and release the patch, disclosure is meaningless. Thus, our final sample consists only of vulnerabilities not discovered by the vendor himself. Of these 1469 observations, 158 observations come from SecurityFocus alone (i.e., CERT/CC did not publish them at all), which are related to 85 vendors and 89 vulnerabilities, 110 observations come from CERT/CC alone (i.e., SecurityFocus did not publish them all), involving 57 vulnerabilities and 54 vendors, while the rest were jointly published by both. The data unique to SecurityFocus

[10] CERT/CC lists these vendors as "status unknown" in their vulnerability publication. Since most of these vendors do not patch, excluding them would bias upwards the estimated coefficient on the CERT dummy. A random sample of 50 vulnerabilities that are disclosed by both CERT and Security Focus indicates that 13% of the vendors listed as status unknown are listed as vulnerable by SecurityFocus. Thus we conclude that the bias implied by excluding these observations is smaller than the bias involved in including them.

allows us to identify the effect of CERT/CC while the joint data allows us to identify the effect of disclosure per se. We deliberately over-sample the joint data to control for unobserved difference between sources.

There are several differences between CERT/CC and SecurityFocus regarding both the publication of a vulnerability as well as its disclosure. While CERT/CC is more cautious in its approach to disclosure (provides about 45 days to vendors), SecurityFocus tends to favor *instant disclosure*. There are other differences as well. (1) SecurityFocus typically publishes all the vulnerability reported, while CERT/CC tends to highlight more significant vulnerabilities; (2) the vulnerability information published by SecurityFocus is from the discussion on the Bugtrag mailing list and vendor statement, while CERT/CC has higher standards on what is eventually published through its notes because of additional research performed by CERT/CC employees; (3) CERT/CC notifies vendors before publishing information and provides them with a disclosure window. Though a benign identifier of a vulnerability may also notify the vendor before posting the vulnerability information on SecurityFocus, SecurityFocus generally does not contact the vendor itself. In contrast, CERT/CC has a well established relationship with many software vendors and is, therefore, able to communicate its information to the right person at the vendor. Overall, compared to SecurityFocus, CERT/CC targets more severe vulnerabilities, has higher standards on which vulnerabilities eventually get published, communicates with vendors regularly and potentially has more influence.

4.2 Disclosure

We measure the patching time as the time elapsed between notification (the relationship between notification and publication is clarified below) and patch availability. For a given vendor vulnerability pair, we consider a vulnerability to be *disclosed* if the vulnerability is made public by anyone (CERT/CC, SecurityFocus or other sources such as other vendors) before the vendor has released a patch. For the same vulnerability, disclosure conditions may vary across vendors. For instance, a vendor may have released the patch 10 days after notification, which would constitute disclosure for other vendors that are also affected by the same vulnerability but have not yet released a patch.

Though SecurityFocus is well known for its instant disclosure policy, not all vulnerabilities published by SecurityFocus are disclosed instantly. Some identifiers inform the vendor first and wait for some time before posting the vulnerability information on SecurityFocus website. By reading through the details of the interaction between the identifier and the vendor, we are able to identify the actual vendor notification date. About 35% of SecurityFocus-only observations involve vendors that were informed by the identifiers prior to the disclosure of information on SecurityFocus.

For vulnerabilities published by CERT/CC, the date of notification is published along with the vulnerability if CERT/CC actually contacted the vendor about the vulnerability. When no information on vendor notification is available, we assume the vendor is notified when the vulnerability is published by SecurityFocus or CERT/CC. While CERT/CC has a stated 45 days disclosure policy, in many instances the vulnerabilities get disclosed much earlier because: (i) Not all vendors are notified at the same time, perhaps because it takes time to determine if a vendor is vulnerable. In some cases, this may result in a vendor being notified after disclosure. (ii) Conversations with CERT/CC staff suggest that a vulnerability could be disclosed before the 45 days period if more than 80% of the vulnerable vendors have patch ready. (iii) While CERT/CC tries to keep the vulnerability confidential during the disclosure window *T*, a vulnerability can be disclosed by others, such as SecurityFocus or a vendor. Table 1 defines disclosure, notification and publication.

Table 1: Definition of Important Terms

Term	Description
Vendor Notification	The event that a vendor is notified of the vulnerability. CERT/CC provides information on when it contacted the vendors about a vulnerability. In case of SecurityFocus (SF), if the identifier contacted the vendor then the vendor notification date is available. Otherwise, vendor notification happens when the vulnerability is published by SF. In case the vulnerability information was made public by other sources before CERT/CC and SF took any action, we consider the earliest known public date as the vendor notification date.
Vulnerability Publication	The event that a vulnerability is made public by CERT/CC or SF. After notifying the vendor, CERT/CC typically waits some time before publishing the vulnerability. SF typically publishes the vulnerability information immediately though the vendors could have been notified earlier by the identifier. A vulnerability may be published by both CERT/CC and SF.

Disclosure	Disclosure happens to a vendor when the vulnerability information is made public before the vendor releases a patch. The source of disclosure includes SF, CERT/CC or third parties. SF and CERT/CC make the information public by publishing vulnerability notes while other sources make the information public in variety of ways.
Patching Time	The time elapsed between patch release date and notification date is the patching time. Disclosure may or may not have occurred during this time. When a patch is not released, this is treated as a censored observation.

4.3 Key Variables and Descriptive statistics

We define the key variables of interest and descriptive statistics below.

(i) *Patching Time*: This is our dependent variable, measured as the number of days elapsed between notification of vendor *i* for vulnerability *j* and availability of the patch. In case a vendors are not notified or notified after publication, the patching time is measured by the number of days elapsed between publication and availability of the patch. The information on when a patch was released was collected from CERT/CC and by visiting vendor's websites. Most the vendors release advisory information publicly on their websites. We take the earliest public date of patch availability as the patching date. Patching statistics are presented in Table 2.

(ii) *Disclosure*: We define disclosure as a dummy variable such that disclosure of vulnerability *i* for vendor *j* at time *t* takes value 1 if the vulnerability *i* is disclosed before time *t* and if the vendor *j* has not released a patch for it, and zero otherwise. Whenever vulnerability information is made public before the patch has been released, the vulnerability has been disclosed.

Table 2: statistics on patching and disclosure

Patching Time (in days)	
Mean	56.5 (114.8)
Median	19
% patched	90%
Disclosure Time (in days) [11]	

[11] Elapsed time between Vendor Notification and Disclosure (in days) is approximately 15.6 days. We refer to this elapsed time as disclosure time.

Mean	15.6 (43.0)
% instant disclosure	67.1%
Number of observation facing disclosure	985

Standard errors are in parenthesis

From table 2, it is important to note that a significant number of vulnerabilities are instantly disclosed in our sample. This seems surprising but many vulnerabilities are publicly announced first, picked up by CERT/CC later and then the patch is released by the vendor. Note that if a vendor has released a patch before the vulnerability is published, for that vendor, disclosure takes a value 0. The number of observations facing disclosure is 985, which involves 364 vulnerabilities and 236 vendors.

(iii) *CERT Effect*: As noted earlier, our data consists of vulnerabilities published by CERT/CC, by SecurityFocus and by both. We also noted that vulnerabilities handled (i.e., eventually published) by CERT/CC are more likely to be patched and patched faster. Therefore, to capture the impact of CERT/CC we include a dummy "*CERT*" which takes value 1 for all vulnerabilities published by *CERT* and 0 for vulnerabilities published by SecurityFocus alone.

Table 3: Impact of CERT on patching speed

For All Observations	Published by CERT/CC	Published by SecurityFocus alone
% patched	93.21%	62.66%
Disclosure Time (see footnote 11)	15.49 (38.06)	16.15 (72.20)
Severity Metric	25.43 (22.06)	8.65 (4.40)
Obs / vuls	1311 / 349	158 /89
For Patched Observations		
Patching time	55.38 (114.09)	70.88 (122.52)

Standard errors are in parenthesis

It is clear from table 3 that vulnerabilities handled by CERT/CC are patched faster. On average if vulnerabilities are patched, then patching happens in about 55 days for CERT/CC and about 71 days for SecurityFocus specific vulnerabilities. Also note that CERT/CC publishes more severe vulnerabilities. The average disclosure time is also smaller than 45 days even for CERT/CC.

(iv) *Disclosure Source*: As discussed (see also Table 1) we distinguish between vulnerability disclosure and vulnerability publication. We summarize patching time conditioned on the disclosure source in Table 4. Again it is clear, that there is a strong "CERT" effect. Disclosure by CERT/CC leads to lower patching time and higher percentage of vulnerabilities patched as compared to disclosures by SecurityFocus or other sources.

Table 4: Impact of first disclosure source on patching time

	CERT/CC	SecurityFocus	Others
All observations			
% patched	96.2%	83.1%	89.6%
Severity Metric	38.1 (24.3)	19.6 (20.7)	17.0 (13.6)
Obs / vuls	157/27	550/233	278/104
For Patched Observations			
Patching speed	23.8 (37.1)	53.9 (125.8)	70.5 (147.3)

Standard errors are in parenthesis.

(v) *Vendor Characteristics*: We include vendor size (as number of employees), firm type dummy (public firm or not) and dummy variable for open source or not. The total number of vulnerable vendors in our sample is 325. However, we were able to find reliable information for only 142 of them. Those vendors for which reliable information is not available, (most of which are small or foreign vendors) we simply code as a dummy variable, *small*, which takes value 1 for observations where firm size is missing. In the empirical analysis, we interact (1-*small*) with *size* when estimating the impact of employee size.[12] The reported statistics in table 5 are averaged over only those vendors whose information is available.

[12] Alternative specifications in which vendors with missing size were given a value of zero yield similar results.

Table 5: Vendor Characteristics (*N* = 142)

	Mean	Std. Dev.
Vendor employee size (in 000's)	17.60	66.12
Public firm	0.34	0.47
Open source	0.23	0.42

(vi) *Severity*: We use the CERT/CC vulnerability metric, which is a number between 0 and 180, as our measurement of vulnerability severity[13]. CERT/CC vulnerability metric is a more comprehensive vulnerability severity measurement than the NVD severity indices (which only takes a value of low, medium or high), since it takes other information, like the range of information available and number of systems at risk, into account instead of focusing only on the technical characteristics of a vulnerability. Our empirical results are similar with using NVD severity indices, but the CERT/CC vulnerability metric should better control of vulnerability severity. Vulnerabilities published by SecurityFocus alone do not have a CERT/CC metric. To solve this problem, we first use the 465 vulnerabilities published by CERT/CC and documented by NVD to create linear projection of log (metric) on the vulnerability technical characteristics, reported in Appendix C. (The vulnerability technical characteristics, whose definition is from NVD, are discussed in appendix B.) Then we fit the metric value for the vulnerability published by SecurityFocus alone using this projection. The fit is around 17%. Table 6 summarizes the severity of the vulnerability measured by CERT/CC vulnerability metric and the number of affected vendors per vulnerability.

Table 6 Vulnerability severity metric (N = 438)

	Mean	*Std. Dev.*	*Min*	*Max*
Vulnerability severity metric	14.82	16.48	0	108.16
Number of affected vendors/vulnerability	8.23	21.53	1	242

We also control for the impact of September 11 events on the software vendors. About 52% of the vulnerabilities included in our sample were published after 11th September 2001.

[13] See http://www.kb.cert.org/vuls/html/fieldhelp#metric for definition of CERT vulnerability Metric.

5. Empirical Strategy

Our goal is to examine the impact exogenous factors including disclosure at time t on patching time, conditional on the vendor not having patched till that time. Recall that disclosure can happen at any time, therefore running a linear regression of patching time on disclosure is misleading because of the definition of disclosure (disclosure can happen only before a patch). Thus we estimate a conditional hazard model which will compare the patching time of two vendors who did not patch until time t but one of them faced disclosure at time t while the other did not.

We use a proportional hazard model, which is widely used in management science and economics (Kalbfleisch and Prentice 2002). In our case, an observation is considered as an event under risk following vendor notification at some time (t_0). The event is considered "failed" when the vulnerability is patched at some time (t). The regressors shift the baseline hazard rate proportionally. The key to note in our case is that *disclosure* is a time varying covariate. For a vulnerability i and vendor j, if disclosure happens at some time t_1 such that $t_0 < t_1 < t$ then, *disclosure* = 0 before t_1 and disclosure = 1 after t_1.

Let $\lambda_i(t)$ denote the hazard rate of vulnerability i at time t conditional on remaining un-patched at time t. Then the proportional hazards model (PHM) is

$$\lambda_i(t, X, \beta) = \lambda_0(t)\exp(\beta X) \qquad (1)$$

Where $\lambda_0(t)$ is the baseline hazard rate at time t; β is the vector of coefficient to be estimated and X is a vector of vulnerability specific or vendor specific characteristics. While disclosure is time varying, the other covariates (see section 4.4) do not change with time.

In order to control for unobserved heterogeneity across vendors, we also consider a frailty hazard model (Vaupel 1979). The frailty hazard model function is specified as follows:

$$\lambda_{ij}(t, X, \beta) = \lambda_0(t)\exp(\beta X + e_j) \qquad (2)$$

where i indexes vulnerability and j indexes vendor and $e_j = \log(z_j)$. The term z is the unobserved frailty assumed to be gamma distributed with mean 1 and variance σ^2 where the variance is to be estimated. The frailty model is akin to "random-effect" model where z is a random variable fixed within the group of interest. In our data we let z remain fixed within vendors. The idea is that vendors may have specific patching policies which affect the vendor's patching decisions but are unobserved by us. We also estimated a specification (not reported here) where *z* was fixed within vulnerabilities but the results are similar to those reported here.

Before proceeding to the estimations, in figure 3 we present the Kaplan-Meier (1958) (KM) estimate of the survivor function or the probability of survival past time *t*, with and without disclosure. It is clear from the figure that disclosure reduces the survival time. Recall that survival in our case is absence of a patch Thus, on average, disclosed vulnerabilities get patched more frequently and faster than non-disclosed vulnerabilities. By 50th day, almost 75% of the vulnerabilities are patched if publication happened before patching and less than 50% are patched if no disclosure took place.

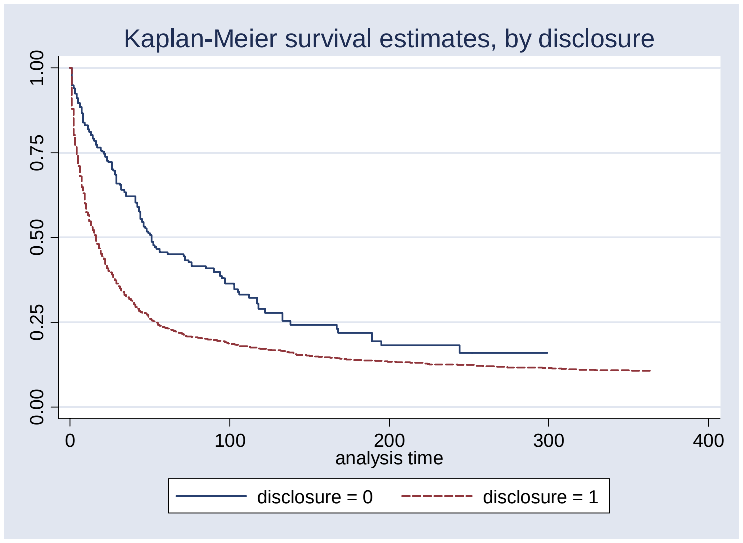


Figure 3: Kaplan-Meier Survival estimates, by disclosure

6. Results and Discussion

There are two ways to estimate the models in (1) and (2). We can either specify the baseline hazard function (λ_0) parametrically (e.g., as a Weibull distribution) or characterize it as non-parametrically (as in the Cox proportional model). Parametric hazard forms are easier to work with, especially for model predictions. Non-parametric forms, however, require fewer restrictions.

We report estimates for models (1) and (2) by maximum likelihood method assuming the baseline hazard has the form of Weibull distribution. The Weibull has a baseline hazard of the form $\lambda_0 = \gamma\ p\ t^{p-1}$. It is a very flexible distribution with two parameters, γ and *p,* and generally tends to fit the data well. Parameter *γ* is the scale parameter and *p* is the shape parameter which determines the shape of the Weibull distribution. We also tried the partial likelihood method with nonparametric baseline hazard suggested by Cox (1975), which yielded very similar results (see appendix D where we report the Cox estimates).

First we estimate the model without distinguishing between disclosure sources. Later we distinguish between various disclosure sources. Estimates with and without frailty models are presented in Table 7.

We report hazard ratios, which are easier to interpret (the hazard ratio is simply exp(*β*)). A hazard ratio less than 1 signifies that the covariate decreases the hazard rate while hazard ratio greater than 1 increases the hazard rate. Since hazard in our case is defined as "hazard of patching", an estimate of 3.04 for CERT means that for the vulnerabilities handled by CERT, the hazard of patching increase by 204% at any time. This is consistent with the observation that CERT/CC tends to publish more serious vulnerabilities and has strong lines of communication with vendors. Thus hypothesis H3 is supported.

Table 7: Estimates (Hazard Ratio) Weibull Hazard Model (N = 1469)

	(1) With frailty		(2) Without frailty	
	Haz. ratio	Std. Error	Haz. ratio	Std. Error
CERT_effect	3.04***	0.35	3.10***	0.33
Disclosure	2.37***	0.20	2.19***	0.17
Small	0.84	0.51	0.80**	0.09
*(1-Small)*size (log)*	1.01	0.03	1.00	0.01
Public firm	1.32	0.26	1.28***	0.11
Open source software	1.71***	0.30	1.61***	0.14
Severity metric (log)	1.24***	0.03	1.20***	0.03
Post 9_11	2.08***	0.14	2.05***	0.13
ln_p (Weibull shape parameter)	-0.58***	0.02	-0.65***	0.02
σ (frailty parameter)	0.32***	0.10		
Log Likelihood	-2978		-3014	

Notes: * indicates significant at 10% level, ** indicates significant at 5% level and *** indicates significant at 1% level.
The constant for the proportional hazard model is not identified when hazard ratio is estimated.

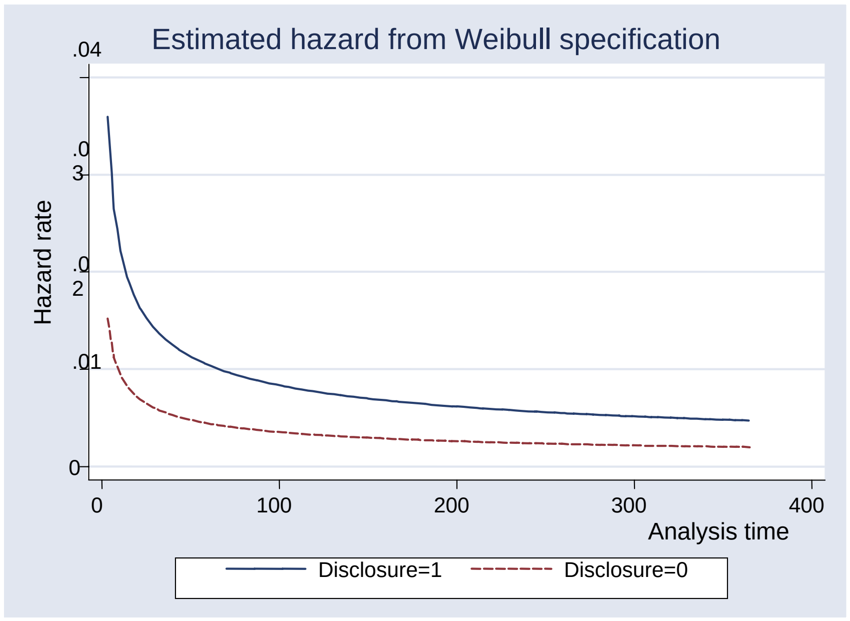
The frailty parameter (σ) is significant indicating that controlling for frailty was important. Note that the parameters estimates are similar in both specifications. The shape parameter (p) for the Weibull is significant as well. Since p is less than 1, this suggests a decreasing hazard (see figure 4 as well).

The key variable of interest is disclosure. The estimated impact of disclosure is large and in the expected direction. The hazard of patching increases by almost 137% when the vulnerability is disclosed than otherwise. Thus, conditional on a vendor not having patched till time t, disclosure at time t increase the hazard of patching by 137%. As table 8 shows, this translates to a patch that comes 25 days faster when the vulnerability is disclosed on the 1st day itself (recall that disclosure is time varying). Thus hypothesis *H1* is supported.

Employee size is insignificant thus hypothesis *H2* is rejected in favor of the null hypothesis of no effect. Once we control for frailty, firm type (publicly traded or not) is insignificant and hence *H4* is rejected. Severity is highly significant and in the expected direction, supporting *H3*. A percent increase in severity metrics increases the hazard of patching by 24%. Open source vendors patch faster than closed source vendors. On an average, open source vendors are 71% more likely to patch at a given time than closed source vendors. This is an unexplored aspect of open source studies. Similarly, post 9/11, a vulnerability is 108% more likely to be patched than before 9/11.

To further understand the impact of disclosure, we plot the hazard rate with and without disclosure in figure 4 (keeping other covariates at their mean). As can be seen, disclosure increases the hazard rate about two fold.

Figure 4: Estimated hazard ratio with and without disclosure



While the estimate on *disclosure* highlights the fact that disclosure increases the pace of patch delivery, we still want to examine how disclosing a vulnerability at different times changes the time to patch. In particular, a policy maker like CERT/CC may be interested in knowing how disclosing vulnerabilities at different times would affect the expected patch delivery times. To accomplish this we calculate the predicted expected patching time with and without disclosure. Since disclosure is a time varying covariate, we calculate these predictions for different disclosure times. The calculations and the formula are derived in appendix A. The results highlighting the impact of disclosure at various times on average patching speed are shown in Table 8.

Table 8: The Estimated Effect of Disclosure (in days)

Disclosure time *T*	Expected patching time if disclosed at time *T* (1)	Expected patching time without disclosure (2)	Effect of disclosure (2) – (1)
0 (Instant disclosure)	33.03	61.77	28.74
1	36.90	61.77	24.87
2	38.60	61.77	23.17
3	39.98	61.77	21.79
4	41.16	61.77	20.61
5	42.21	61.77	19.56

6	43.18	61.77	18.59
7	44.07	61.77	17.70
8	44.90	61.77	16.87
9	45.69	61.77	16.08
10	46.43	61.77	15.34

These calculations are done setting the covariates at their sample means, namely that the vulnerable vendor is a public firm and has the average employee size of our sample; the vulnerability is published after 9/11 event, handled by CERT/CC and has the average severity metric of our sample.

Thus, for a typical vulnerability, if a policy maker chooses never to disclose the vulnerability, then on average the patch will come in about 62 days. However, if the vulnerability is disclosed immediately then patch would come in approximately 33 day. Thus, instant disclosure will hasten the patch by almost 29 days. Of course, this does not mean that instant disclosure is the optimal policy. But our results provide policy maker a decision tool where some kind of "what-if" analysis can be performed and the expected impact of various disclosure policies can be understood and measured. This is the key contribution our paper.

Finally, we re-estimate the model but with the disclosure source included as well. Recall that in our data, a vulnerability could be initially disclosed by CERT/CC, SecurityFocus or the third party even though the vulnerability is eventually published by CERT/CC and/or SecurityFocus. To understand the impact of these sources, we create four dummies: C_C - published by CERT and first disclosed by CERT, C_S - published by CERT and first disclosed by SecurityFocus, C_O - published by CERT and first disclosed by another party, S_S - published by SecurityFocus and first disclosed by SecurityFocus. Two cases, namely S_O and S_C are not present in our sample: A vulnerability first disclosed by CERT is also published by CERT and hence it is subsumed in the term C_C. Thus the reference category is all vulnerabilities that are not disclosed (i.e. all those cases when vendors patched before disclosure). We present the results with and without frailty in Table 9.

Table 9: Estimates (Hazard Ratio) with Disclosure source (N = 1469)

	(1) With frailty		(2) Without frailty	
	Haz. Ratio	Std. Error	Haz. ratio	Std. Error
C_C	1.79***	0.18	1.58***	0.15
C_S	1.19**	0.09	1.22***	0.09
C_O	1.05	0.09	1.06	0.09
S_S	0.30***	0.05	0.29***	0.04
Small	0.79	0.45	0.75***	0.09
*(1-Small)*size (log)*	1.00	0.03	0.99	0.01
Public firm	1.24	0.23	1.26***	0.11
Open source software	1.72***	0.29	1.58***	0.14
Severity metric (log)	1.20***	0.03	1.16***	0.03
Post 9_11	1.66***	0.12	1.69***	0.11
ln_p (Weibull shape parameter)	-0.55***	0.02	-0.62***	0.02
σ	0.29***	0.08		
Log Likelihood	-3021		-3057	

Notes: * indicates significant at 10% level, ** indicates significant at 5% level and *** indicates significant at 1% level.

We find that, as expected, vulnerabilities initially disclosed by CERT/CC have the smallest patching time. Note that all vulnerabilities which are eventually published by CERT/CC (irrespective of who disclosed it) are patched faster (First three dummies are greater than 1). Also see that estimate of C_S suggests that disclosure by SecurityFocus is also potentially significant. They are patched faster than the vulnerabilities not disclosed at all. However the key distinction between C_S and S_S is that in the latter case, the vulnerability is never published by CERT/CC. Note that S_S dummy has the smallest coefficient and is less than 1. This means that vulnerabilities disclosed and published by SecurityFocus alone are taken less seriously. Thus we again find evidence that CERT/CC is an influential source of vulnerability information and publication as well as disclosure by CERT/CC has expected impact on vendor behavior. Thus *H6* is supported. Other estimates are also consistent with results in Table 7.

Robustness

Note that our results are not sensitive to the assumption of the Weibull distribution since the Cox non parametric specification yields similar results. Further, controlling for the source of disclosure, and for unobserved heterogeneity in vulnerabilities also does not qualitatively affect the results.

We also controlled for vulnerability specific random effect (rather than vendor specific random effects) and find that the results remain unchanged. We also conducted sensitivity analysis around patching time by including only patched observations or dropping patching time of more than 1 year. The impact of disclosure and CERT/CC remains qualitatively unchanged. In short, we find that both the CERT_effect and *disclosure* are quite prominent and significant and unlikely to be the artifact of modeling technique or data collection strategies.

The applicability of proportional hazard model critically depends on the proportionality assumption. In particular, proportionality assumption means that the impact of a covariate on the hazard rate is constant irrespective of time. Thus hazard rate shifts proportional up or down through-out the whole duration. If a covariate takes discrete values (like 0 or 1) then one can simply plot log(-log(survival rate)) versus log of survival time. If the two lines are parallel then the proportionality assumption holds. More formally, the proportionality assumption is tested using Schoenfeld residuals. We test for a non-zero slope in a generalized linear regression of the scaled Schoenfeld residuals on functions of time (Therneau and Grambsch 2000; p 127-142). The test for two key variables in our analysis (disclosure and CERT effect) is reported below.[14]

Table 10: Test of Proportionality Assumption

	rho	Chi^2	Prob>chi2
CERT_effet	0.06081	5.38	0.0204
Disclosure	-0.03385	1.74	0.1866

It appears that CERT effect is non proportional (*p* value is less than 0.05). Thus, with time the effect of CERT/CC on patching time changes. One way to correct for lack of proportionality is to interact the CERT dummy with time. We find that the interaction term is positive and significant (see Appendix E) so that the effect of CERT/CC increases with time. We re-ran the proportionality test after including the interaction term between the CERT dummy and time and

[14] This test is available only with Cox models in Stata and that is what we report here.

found that the Schoenfeld residual test cannot reject the proportionality hypothesis for the CERT effect.

7. Conclusions

Vendors' patching behavior is an under-investigated research domain. In recent times, vulnerability disclosure has been a controversial topic and vendors, policy makers and software users have not found consensus about when and how to disclose the vulnerability. Vendors argue for more time and sometimes take steps such as using the courts to stop vulnerability information disclosure. Policy makers like CERT/CC need to decide what the reasonable time is for vendor to come up with a patch. Unfortunately, there is no rigorous empirical research that can shed light on how disclosure in particular and other key factors (like vendors characteristics, vulnerability characteristics) in general condition vendors' patching time. Our paper provides critical empirical estimates on vendor response. To our knowledge, this is the first systematic empirical examination of this question. In particular, a policy maker like CERT/CC can use these results to decide how actually disclosing the vulnerability at any given time will affect the expected patching time of the vendor, given that they have not patched until then.

From a unique data set that we collected, we estimate a proportional hazard model of patching time. We find that disclosure forces the vendors to patch faster. This verifies earlier conjectures that vendors respond to disclosure. In particular we find that disclosure increases vendors' patch delivery speed by almost 137%. Put differently, compared to no disclosure, instant disclosure will force a vendor to release the patch 29 days earlier. We also find that vendors respond slower to vulnerabilities not handled by CERT/CC. This might reflect unmeasured differences in the severity and importance of vulnerabilities. It might also reflect the stronger lines of communication between CERT/CC and vendors, and the value of the vulnerability analysis by CERT/CC.

Open source vendors are more likely to patch when vulnerability is found in their products. Given the attention on the open source vs. closed 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 2005). Severe vulnerabilities are patched sooner than less severe ones, consistent with a rational model in which vendors internalize some of the customers' losses.

While we find that disclosing information always leads to quicker patch, this does not mean that instant disclosure is optimal. To devise an optimal policy, one also needs to understand how hackers' propensity to attack changes with disclosure, and the resulting changes in the extent of customer losses. Some early work by Arora, Krishnan, Nandkumar and Telang (2004) is promising but more empirical work is needed for coherent disclosure policy.

While our results are interesting, there are qualifications. Particularly, our model doesn't control for the quality of the patch. Additional resources allocated to patch development could result in a higher quality patch than a patch that is released sooner. We test the impact of actual disclosure on vendor behavior but not the impact of disclosure window. Future work which can distinguish the impact of actual disclosure from disclosure window will be very useful for policy makers. Our model is also reduced form in nature. Future work can estimate more precise structural models. Given the importance of these issues and the paucity of empirical work, we hope that our study paves the way for more research with new and possibly better data sources.

REFERENCES

Anderson, R., (2001), "Why Information Security is Hard," *In Proceedings of the 17th Annual Computer Security Application Conference*, New Orleans LA. available at http:// www.ftp.cl.cam.ac.uk/ftp/users/rja14/econ.pdf.

Arbaugh, W. A., W. L. Fithen, and J. McHugh (2000), "Windows of vulnerability: A case study analysis," *IEEE Computer*.

Arora, A., Caulkins, J.P., and Telang R. (2005), "Sell first, fix later: Impact of patching on software quality," *Management Science*, Forthcoming.

Arora A., R. Telang, and H. Xu (2004), "Timing Disclosure of Software Vulnerability for Optimal Social Welfare," *The third Workshop on Economics of Information Systems*. Minneapolis, MN.

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

Banker R., G. Davis, and S. Slaughter (1998), "Software Development Practices, Software Complexities, and Software Maintenance", *Management Science*, 44:4, 433-450.

Camp, L. & Wolfram, C. (2000), "Pricing Security" In *Proceedings of the CERT Information Survivability Workshop, Boston*, MA Oct. 24-26. pp-31-39.

Cavusoglu H., Cavusoglu H., Raghunathan S. (2004), "How should we disclose Software vulnerabilities?", *14th Annual Workshop on Information Technologies and Systems*, Washington D.C.

Choi J.P., C. Fershtman, & N. Gandal (2004), "Internet Security, Vulnerability Disclosure, and Software Provision", *The Fourth Workshop on Economics of Information Systems*. Boston, MA.

Cox, David R. (1975), "Partial likelihood", *Biometrika*, May/Aug., 62(2), pp. 269-76.

Chen, P., G., Kataria and R. Krishnan (2005), "Software Diversity for Information Security", *The Fourth Workshop on Economics of Information Systems*. Boston, MA.

Gordon, L.A. & Loeb, M.P. (2002). "The Economics of Information Security Investment". *ACM Transactions on Information and System Security,* 5.

Gordon. S and R Ford (1999) "When Worlds Collide: Information Sharing for the Security and Anti-virus Communities", *IBM research paper.*

Information Week (2005), "Cisco Details IOS Vulnerability Spilled At Black Hat" July 29, 2005. http://www.informationweek.com/story/showArticle.jhtml?articleID=166403842

Kalbfleisch, J.D., and Prentice, R. L. (2002), *The Statistical Analysis of Failure Time Data*, 2nd edition. John Wiley and Sons

Kannan K., R. Telang (2005), "Market For Software Vulnerabilities? Think Again", *Management Science*, 51(5), 726-740.

Kaplan, E. L. & Meier, P. (1958), "Nonparametric estimation from incomplete observations", *Journal of the American Statistical Association*, 53, 457-48.

Krishnan, M. S.; Kriebel, C.; Kekre, S.; & Mukhopadhyay, T (200). "An Empirical Analysis of Cost and Conformance Quality in Software Products." *Management Science,* 46: 745-759.

Nizovtsev D., & M. Thursby (2005), "Economic Analysis of Incentives to Disclose software Vulnerabilities", *The Fourth Workshop on Economics of Information Systems*. Boston, MA.

Ozment, A. (2004), "Bug Auctions: Vulnerability Markets Reconsidered," *The Third Workshop on Economics and Information Security*. Minneapolis MN.

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

Telang R., S. Wattal (2005), "Impact of Vulnerability Disclosure on Market Value of Software Vendors: An Empirical Analysis", *The Fourth Workshop on Economics of Information Systems*. Boston, MA.

Therneau T.M., P. M. Grambsch (2000), *Modeling survival data: extending the Cox model***,** New York, Springer.

Vaupel, J. W., Manton, K.G., & Stallard, E. (1979), "The impact of heterogeneity in individual frailty on the dynamics of mortality", *Demography*, 16, 439-454

Appendix

A: Calculation of the estimated effect of disclosure:

For a Weibull distribution, the hazard function is $\lambda(t,\beta,x) = \exp(x\beta)\lambda_0$, where $\lambda_0(t,\gamma,p) = \gamma \cdot p \cdot t^{p-1}$ is the baseline hazard function. Since *F*(*t*) = *f*(t)/λ(t) where F(.) and f(.) are the associated distribution and density functions, we can solve this as first order differentiation equation to get F(t) or the probability of patching as $F(t,\gamma,p,\beta,x) = 1 - \exp(-\exp(x\beta)\cdot\gamma\cdot t^p)$. With the estimated parameter β, γ and p, we can construct F (t | disclosure =1, $x = X$) and F(t | disclosure =0, $x = X$) for any give observation *X*. Ignoring *X* for convenience of notation, we define

P(t| T) = Pr (patch before t | disclosure at T)

K(t) = Pr (patch before t | disclosure = 0) = F(t| disclosure = 0)

G(t) = Pr (patch before t | disclosure = 1) = F(t| disclosure = 1)

$$\text{Then } P(t \mid T) = \begin{cases} \text{K (t) if } t \le T \\ (1-K(T))\left[\dfrac{G(t)-G(T)}{1-G(T)}\right] + K(T) \text{ if } t > T \end{cases}$$

For $t \le$ T, the patch comes before disclosure. For $t >$ T, the probability that a patch arrives in the interval [0, T] is K(T). The probability that it arrives in (T, *t*] is given by the first term, in which (1-K(T)) is the probability of not patching till T and the term multiplying it is the probability of patching at by *t* conditional on not having patched till T,. It can be verified that P(.) is a well defined distribution function, because P() is monotonically increasing in *t*, P(0) = K(0) = 0, P(∞) = 1, and P() is continuous everywhere on its domain.

To calculate expected number of days, we construct the discrete version of the probability density function. The probability of patching on day *t* conditional on disclosure at T is equal to P(t | T) - P(t – 1 | T). Then expected number of days for patch conditional on disclosure at time E(t/T) = $\sum_{t=1}^{N} t[P(t/T) - P(t-1/T)]$. Since we know all values and distribution functions, we can calculate E(.) for various values of disclosure time T.

B: Vulnerability technical characteristics (N=303)

Variable	Description	Mean	Std. Dev.
AR_LR	Remote attachable. Attacks that utilize the vulnerability can be launched across a network against a system without the user having previous access to the system.	0.76	0.43
LT_OAP	Obtain all privilege. Vulnerabilities enable an attack that can gain all the privilege of a system.	0.25	0.43
LT_C	Lose Confidentiality. Vulnerabilities enable an attack that can directly steal information from a system.	0.14	0.35
LT_I	Lose Integrity. The vulnerability enables an attack that can directly change the information residing on or passing through a system.	0.17	0.37
VT_IVE	Boundary overflow. Vulnerabilities, when the input being received by a system, be it human or machine generated, cause the system to exceed an assumed boundary thereby causing a vulnerability.	0.54	0.50
VT_BO	Buffer overflow. Vulnerabilities caused by input being received by a system that is longer than the expected input length. If the system does not check for this condition then the input buffer fills up and overflows the memory allocated for the input. By cleverly constructing this extra input, an attacker can cause the system to crash or even to execute instructions on behalf of the attacker.	0.30	0.46
VT_ECE	Exceptional condition handling error. Vulnerabilities caused by an exceptional condition that has arisen. The handling (or mishandling) of the exception by the system enables a vulnerability.	0.10	0.30
VT_DE	Design Error. Vulnerability is characterized as a "Design error" if there are no errors in the implementation or configuration of a system, but the initial design causes a vulnerability to exist.	0.23	0.42
EC_SA	Server Application. Vulnerabilities occurred in an application providing services to the other users or computers on the network.	0.56	0.50

The definition of the technical characteristics is from NVD at http://icat.nist.gov/icat_documentation.htm.

C: Log linear regression: Dependent Variable =Log (Metric), N=465

	Coef.		Std. Err.
NVD Severity index	0.44	***	(0.13)
AR_LR	-0.63	***	(0.17)
LT_OAP	-0.21		(0.16)
LT_C	0.19		(0.18)
LT_I	0.26		(0.19)
VT_IVE	-0.04		(0.17)
VT_BO	-0.37	*	(0.18)
VT_ECE	-0.27		(0.22)
VT_DE	0.06		(0.16)
EC_SA	-0.09		(0.13)
CONSTANT	0.35		(0.37)
R^2	0.17		

D: Estimates (Hazard Ratio) of PH Model, using non-parametric Cox model

	(1)		(2)	
	With frailty		Without frailty	
	Haz. ratio	Std. Error	Haz. ratio	Std. Error
CERT	2.68***	0.30	2.66***	0.29
Disclosure	2.56***	0.21	2.48***	0.19
Small & foreign vendor	0.95	0.36	0.85	0.10
Employee size (log)	1.01	0.02	1.00	0.01
Public firm	1.26	0.19	1.22**	0.11
Open source software	1.60*	0.22	1.52***	0.13
Severity metric (log)	1.18***	0.03	1.16***	0.03
Post 9_11	1.73***	0.12	1.69***	0.11
Σ	0.12***	0.05		
Log Likelihood	-8459		-8469	
N	1469		1469	

Notes: * indicates significant at 10% level, ** indicates significant at 5% level and *** indicates significant at 1% level.

E: Estimates (Hazard Ratio) of PH Model, using non-parametric Cox model and including interaction variable CERT*t

	(1) With frailty		(2) Without frailty	
	Coefficient	Std. Error	Coefficient	Std. Error
CERT	0.838***	0.129	0.831***	0.127
Disclosure	0.940***	0.080	0.905***	0.078
Small & foreign vendor	-0.058	0.384	-0.164	0.114
Employee size (log)	0.008	0.021	0.000	0.012
Public firm	0.236*	0.147	0.200**	0.088
Open source software	0.469***	0.139	0.416***	0.086
Severity metric (log)	0.170***	0.026	0.153***	0.025
Post 9_11	0.549***	0.067	0.529***	0.064
*CERT*t*	0.002*	0.001	0.002*	0.001
σ	0.122***	0.054		
Log Likelihood	-8456		-8467	
N	1469			

Notes: * indicates significant at 10% level, ** indicates significant at 5% level and *** indicates significant at 1% level.
The estimated is the coefficient, not the hazard ratio.