

# **An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure<sup>1</sup>**

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*

## **Abstract**

A key aspect of better and more secure software is timely patching of the vulnerabilities by software vendors in their products. Software vulnerability disclosure, which refers to the publication of vulnerability information before a patch has been issued, has generated intense debate. An important consideration in this debate is the behavior of software vendors. How quickly do vendors patch vulnerabilities, and how does disclosure affect patch release time? This paper compiles a unique data set from CERT and SecurityFocus to answer this question. Our results suggest that disclosure accelerates patch release. The instantaneous probability of releasing the patch rises by nearly two and a half times due to disclosure. Open source vendors release patches more quickly than closed source vendors. Vendors are more responsive to more severe vulnerabilities. We also find that vendors respond more slowly to vulnerabilities not disclosed by CERT. We verify our results by using another publicly available dataset and find that results are consistent. We also show how our estimates can aid policy makers in their decision making.

**keywords:** Security vulnerability, disclosure policy, patch-release time, open source vendors, information security.

Forthcoming: *Information Systems Research*

---

<sup>1</sup> Author names are in alphabetical order. Authors would like to thank the staff of CERT for the data as well as valuable insight into disclosure process. We also thank participants of WEIS'05 (Workshop on Economics of Information Security) Boston; WISE'05 (Workshop on Information Systems and Economics) Irvine; University of Pittsburgh, University of Texas, Austin, American University and University of Maryland, College Park for their useful feedback. An earlier version of this paper is presented in ICIS, 2006 Milwaukee. Authors acknowledge generous financial support of Cylab, Carnegie Mellon University. Rahul Telang acknowledges the generous support of National Science Foundation (NSF) through the CAREER award CNS-0546009.

## 1. Introduction

Information security breaches pose a significant and increasing threat to national security and economic well-being. According to the Symantec Internet Security Threat Report (2003), firms experienced an average of about 30 attacks per week. Anecdotal evidence suggests that losses from cyber-attacks can run into millions of dollars. The CSI-FBI survey (2005) estimates that the loss per company was more than \$500,000 in 2004 and more than \$200,000 in 2005.

These cyber-attacks often exploit software vulnerabilities. Over the last few years, the number of vulnerabilities found and disclosed has increased dramatically. The Computer Emergency Response Team/Coordination Center (CERT) has published more than 2000 security bulletins for operating systems related vulnerabilities in 2005 alone and has reported more than 82,000 incidents involving various cyber attacks in 2003. 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 the timely release of patches by vendors for the vulnerabilities in their products. Patch release can be viewed as post sales product support. The following quote sums this up<sup>2</sup> –

*“Developing and deploying patches is an increasingly important part of the software development process” – Joseph Dadzie, Microsoft Corporation, March 2005.*

However, while vendors’ incentives to release timely, high quality software has been well studied (Krishnan et al 2000), their patch release behavior is an under-investigated component of overall software quality and security (Arora, Caulkins, Telang 2006). The issue of patches has also gained prominence because of the public disclosure of vulnerabilities. Typically, once a vulnerability is made known to the vendor, the vendor is expected to release a timely patch. However, not all see the vulnerability disclosure process as working well. Many users believe that vendors are not responsive enough and that their patches are of poor quality. As a result, some in the IT community disclose vulnerability information upon discovery, even before the patch is out. The argument is that quick disclosure increases public awareness, makes public the

---

<sup>2</sup> <http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=287>. Accessed 19 June, 2005.

information needed for users to protect themselves, puts pressure on the vendors to issue patches quickly, and, 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 vulnerability disclosure, especially without a good patch, is dangerous for it leaves users defenseless against attackers.<sup>3</sup> 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."<sup>4</sup>

Vulnerability disclosure is a critical issue for software vendors, IT managers, and policy makers who have to devote significant resources to release patches, implement patches, and manage information dissemination respectively. CERT, an influential non-profit organization, and probably the most important conduit for disclosing vulnerabilities, favors a cautious approach to vulnerability disclosure. After learning of a vulnerability, CERT contacts the vendor(s) and provides a time window, which we refer to as the "protected period", within which to release the patch for the vulnerability. The declared policy (and the de facto policy since October 2000) is to give vendors a 45 day "protected period". Typically, the vulnerability is disclosed when the period ends 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 a consortium of 11 software vendors, has suggested its own guidelines (mainly that the vendors be given 30 days).<sup>5</sup> 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 are theoretical models of how disclosure policies affect software vendors' patch release behavior (see Arora, Telang and Xu 2007 for details), there is little empirical research on vendors' patch release behavior in general and of the impact of disclosure in particular. A sensible disclosure policy is possible only if we can quantify vendors' response to disclosure.

---

<sup>3</sup> 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 for this vulnerability were disclosed at a 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 a significant threat against existing infrastructure currently deployed". See <http://www.informationweek.com/story/showArticle.jhtml?articleID=166403842>

<sup>4</sup> See [http://www.blackhat.com/html/bh-usa-02/bh-usa-02-speakers.html#Richard Clarke](http://www.blackhat.com/html/bh-usa-02/bh-usa-02-speakers.html#Richard%20Clarke).

<sup>5</sup> Members include @stake, BindView, Caldera International (The SCO Group), Foundstone, Guardent, ISS, Microsoft, NAI, Oracle, SGI, and Symantec. For more details see <http://www.oisafy.org>

Thus the major goal of our paper is to empirically estimate (i) the key factors that affect vendors' patch release decision, and (ii) whether actual disclosure induces vendors to release the patch faster, and by how much.

We compile a unique dataset of 420 vulnerabilities, published by National Vulnerability Database (previously the NIST ICAT METABASE) from September 26, 2000 to August 11, 2003. Since a vulnerability may involve more than one vendor, our empirical analysis is based on 1429 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,<sup>6</sup> if and when the vulnerability was publicly disclosed, the nature and severity of the vulnerability, and vendor characteristics.

We find that *disclosure* increases the vendor's patch release speed considerably: A vendor is nearly two and a half times more likely to patch after disclosure than before. In particular, if a vulnerability is disclosed instantly (i.e., at the same time that the vendor is informed of it) then, on an average, the patch comes 35 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 disclosed by CERT. This potentially reflects the stronger lines of communication between CERT and vendors, the value of the vulnerability analysis by CERT, and, more importantly, the reputation of the source in interacting with the vendor. We also do an external validity check by using data collected by another source and find that results are consistent with our estimates.<sup>7</sup>

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

---

<sup>6</sup> 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 some cases, the customers need to purchase and install a new version.

<sup>7</sup> Brian Krebs of The Washington Post collected and posted data on Microsoft patching times on his web site ([http://blog.washingtonpost.com/securityfix/2006/01/a\\_time\\_to\\_patch.html](http://blog.washingtonpost.com/securityfix/2006/01/a_time_to_patch.html) )

## 2. Literature review

Our work draws from multiple literature streams. The first is the literature on software quality. Krishnan et al (2000) show that having better personnel, making more use of CASE tools, and more upfront investments in planning and design improve product quality. However, patch release behavior is not studied. Banker, Davis and Slaughter (1998) focus on ex-post software support, namely software maintenance, and examine how complexity and programmer capabilities affect maintenance costs. In this paper, we estimate how vendors respond to disclosure in terms of how quickly they release a patch.

The second stream of work we draw from is the literature on economics of information security. Anderson and Moore (2006) provide a good overview of the issues related to economics of information security, including software vulnerabilities. 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 (2005) employ a formal model to examine whether a market based mechanism is better than the setting in which a public agency like CERT acts as an intermediary. They show that markets always perform worse than CERT because privately optimal disclosure rules are socially sub-optimal. Ozment (2004) analyzes, along the same lines, how auctions perform in such markets.

Recently, some attention has been paid to issues related to vulnerability disclosure and vendor patch release. Choi, Fershtman and Gandal (2005) model a firm's decisions on upfront investment in the quality of the software to reduce potential vulnerabilities, whether to announce vulnerabilities, and whether to apply a patch, and a rational consumer's decision on whether to purchase the software. Nizovtsev and Thursby (2007) model the incentives of users to disclose software vulnerabilities through an open public forum, and derive conditions under which instant disclosure is socially optimal. August and Tunca (2007) examine the role of patching by end users and how unpatched users exert externalities on other users. Png, Tang and Wang (2006) model a game between users and attackers. They show that externalities cause users to underinvest in security and suggest policy measures to remedy the problem. Telang and Wattal (2007) 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.

Our paper is more directly related to Arora, Telang, and Xu (2007) and Cavusoglu et al (2005), who analyze vulnerability disclosure policy. In both, the threat of disclosure conditions when the vendor releases a patch. However, our focus in this paper is on the impact of disclosure itself, rather than on the threat of disclosure. Another key distinction of our paper is that most of the work in the area of economics of information security is analytical and theoretical in nature. In contrast, our paper brings data to some of these theories.

### **3. Vendor's Patch Release Decision**

A software vendor makes two decisions in the patch release process. First, it decides the initial patch release time when it learns about the vulnerability. At some point, the vulnerability may be disclosed. If the vendor has not come up with a patch by the disclosure time, then it needs to make a second decision on whether and how much to accelerate the patch release. The influence of disclosure, the subject of this paper, refers to the second decision. More precisely, we estimate if and how disclosure affects patch release time relative to the baseline (the first decision).

Fully characterizing a software vendor's patch release decision, a stochastic dynamic programming problem, is beyond the scope of this paper, whose focus is primarily empirical. However, the underlying intuition of how disclosure and other factors condition the vendor's patch release decision can be illuminated with a simple model. In this model, we compare the vendor's incentives to release the patch in two cases, one where the vulnerability has been disclosed and the other where it *may* be disclosed in the future.

The vendor faces two types of costs in deciding when to release the patch. The first 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 the patch release time is  $\tau$  (measured from the time when the vendor was informed of the vulnerability), the cost of patch development is  $C(\tau; Z)$ , where  $Z$  represents vendor characteristics, such as size. We expect that  $C(\tau)$  is decreasing in  $\tau$ .

Second, the vendor's customers suffer losses if that vulnerability is discovered and exploited by attackers. Let the expected total customer loss be represented by a function  $L(\tau; X)$  where  $X$  are exogenous factors that condition customer loss, such as the vulnerability severity. Unlike other products, the product liability laws do not apply to software. Thus, we assume that a vendor internalizes a fraction,  $\lambda$ , of customer losses potentially due to loss of reputation, loss of future sales, and contractual service obligations. The vendor chooses the patch release time,  $\tau$ , to minimize

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

The analysis for two polar cases is helpful. First, if the vulnerability has not been disclosed, attackers may discover the vulnerability on their own, or a third party (researchers, users or even other vendors) may disclose the vulnerability at a future date  $z$ .<sup>8</sup> The vendor releases the patch because he is *threatened* with disclosure in the future. In this case,  $L(\tau, X) = E_z [l(\tau-z; X)]$ , where  $l(\tau-z; X)$  is the loss suffered by users if attackers discover the vulnerability at time  $z$ . Since  $z$  is uncertain, we take expectations over  $z$ .

Now consider the second case where the vulnerability has been disclosed at time 0. Disclosure will reveal the vulnerability to attackers with certainty. In this case,  $L(\tau, X) = l(\tau, X)$ , there is no uncertainty unlike the first case. We show in Appendix A that, all else equal, the expected total customer loss will be higher if the vulnerability has been disclosed than if the attackers have to find it for themselves (as is the case when disclosure is uncertain). We also show that the *marginal* benefit of releasing the patch will be higher when the vulnerability has been disclosed, than when the vulnerability is still a secret. It follows that the vendor will release the patch faster when the vulnerability has been disclosed than otherwise. This intuition should carry over to the more realistic, but also more complicated, setting where the vendor can adjust the target patch release time as events unfold.

To anticipate our empirical exercise, we expect that the instantaneous probability of releasing the patch should increase after disclosure. Formally, we state

*H1: Disclosure reduces the software vendor's patch release time.*

---

<sup>8</sup> Disclosure increases customer loss  $L(\tau, X)$  because disclosure makes it easier for the attackers to find the vulnerabilities (see Arbaugh et al 2000; Arora, Nandkumar and Telang 2006 for details).

Next we briefly discuss the factors that condition the baseline rate of patch release. We hypothesize (and the Appendix A sketches out the proof) that factors that increase patch development costs should increase the patch release time and the factors that increase customer losses should decrease the patch release time. In the following, we identify some key factors that affect a vendor's patch release time. Note that we do not actually observe  $C$  and  $L$  and hence our model should be interpreted as reduced form in nature.

*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 vulnerability disclosure. 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 a higher  $\lambda$ . Large vendors may also release the patch faster if they are better able to afford larger patch development teams. Since patch development costs do not vary significantly with the number of software users, the model suggests that large vendors should release the patch faster than the smaller vendors (both due to higher  $L$  and higher  $\lambda$ , and perhaps lower marginal cost of patch release). Thus we hypothesize

*H2: Larger vendors release patches faster than the smaller vendors.*

*Severity of Vulnerability:* The severity of a vulnerability is a multi-faceted concept. The availability of an exploit tool or exploit code, the ability to exploit the vulnerability remotely, the number and importance of systems affected by the vulnerability, and the level of control gained by a successful exploit; all of these imply a higher severity. It is plausible that each of these factors increases the loss users suffer. It is also very likely that loss per unit time that users are exposed is higher for more severe vulnerabilities. It follows that severe vulnerabilities necessitate quicker patches.

*H3: Vendors release patches more quickly for more severe vulnerabilities.*

There is ample evidence that the credibility of information depends upon its source and also upon who validates it. It is likely that vendors respond more quickly to vulnerabilities disclosed by CERT because it enjoys a strong reputation and its pronouncements have more visibility with customers as well. In part, CERT's pronouncements are taken more seriously than other sources because CERT staff investigates the vulnerability before contacting vendors. Identifying this



CERT effect is tricky because it is plausible that vulnerability specific unobserved effects may also be correlated with whether CERT handles (publishes) the vulnerability. For example, CERT may handle only important vulnerabilities. To capture the credibility and visibility that CERT has in the community, we focus on the vulnerability published by both CERT and SecurityFocus (in our sample, a majority of the vulnerabilities are published by both). Within this set of vulnerabilities *published* by both CERT and SecurityFocus, we compare patch release times for vulnerabilities disclosed by CERT to those disclosed by SecurityFocus or other parties. Thus our approach controls for the potential selection bias that arise from unobserved heterogeneity. We expect that

*H4: Vendors respond faster to the vulnerability disclosed by CERT than by other parties.*

*Open Source Vendors:* A stream of work has argued that the open source vendors provide better quality and are more responsive to customers (Wheeler, 2002). A typical argument presented is that since open source code is viewed and reviewed by many, it is inherently of good quality. Similarly, bug reports in open source projects are also acted upon quickly because many users are constantly paying attention to them. Thus, we hypothesize that

*H5: Open source vendors release patches faster than the closed source vendors.*

## **4. Data and variables**

### **4.1 Data sources**

The vulnerabilities studied in our research are from the two most important sources of information on vulnerabilities, CERT and SecurityFocus. CERT publishes information on a wide variety of software vulnerabilities in the form of "CERT Vulnerability Notes", whereas SecurityFocus hosts a well known full disclosure mailing list, Bugtraq, for the detailed discussion and announcement of software vulnerabilities. Both CERT and SecurityFocus cross-reference their vulnerability databases with the National vulnerability Database (NVD) through the Common Vulnerabilities and Exposures (CVE) catalog.<sup>9</sup> The National Vulnerability Database tracks a large number of security problems, but not all CERT and/or SecurityFocus vulnerabilities meet its criteria of being listed in the database. For the purpose of our empirical

---

<sup>9</sup> The CVE name is the 13 character ID used by the "Common Vulnerabilities and Exposures" group to uniquely identify a vulnerability.

analysis, we consider only the NVD listed vulnerabilities that are published by CERT and/or SecurityFocus. Since a vulnerability may affect many products, our unit of observation is a vendor-vulnerability pair. We augment these data by adding information about whether and when a patch was released, using a variety of sources including the vendor websites. We also collect data on vendor size and other characteristics, from the vendor's website and from Hoover's online database.

## **4.2 CERT vs. SecurityFocus**

The vulnerability publication of CERT and SecurityFocus follows different process. Upon receiving a vulnerability report from an identifier, CERT researches the vulnerability and, for vulnerabilities that meet its criteria, contacts the affected vendor(s) and coordinates patch development before making the vulnerability public by publishing it. On average, in a year, about 3000 vulnerabilities get reported to CERT, of which only about 10 percent are published. CERT reports a list of affected vendors for each vulnerability. Additional summary information is provided for each vendor as well, including the date when CERT notified the vendor.

A vendor typically responds to the CERT vulnerability notification in one of the following forms: (i) acknowledging the presence of the vulnerability and collaborating with CERT to release a patch (ii) contending that the product(s) in question is not vulnerable, in which case CERT just lists the vendor as not vulnerable or as vulnerable without a patch. In case a vendor chooses not to respond, CERT records the vendor as vulnerable without a patch or vulnerable status unknown, as appropriate.

SecurityFocus does not require advance notification to the affected vendors before the vulnerability is made public. However, the discoverer of the vulnerability may choose to notify the vendor before reporting it to SecurityFocus. Our data show that in about 35 percent of cases when a vulnerability is published by SecurityFocus, the identifiers informed the vendors prior to the disclosing it on SecurityFocus. Furthermore, the identifiers tend to include the vendor notification date and sometimes the notification itself into their vulnerability reports to SecurityFocus.<sup>10</sup> When a vulnerability is published by SecurityFocus, the affected vendors can

---

<sup>10</sup> Some of the original postings of the discoverers have recently been removed by SecurityFocus from the vulnerability web pages but were available at the time we collected the data.

post their solutions to the Bugtraq mailing list. Sometimes, a reference to the vendors' solution published elsewhere is posted to Bugtraq by others.

To summarize, there are three differences between CERT and SecurityFocus regarding both the publication of vulnerability as well as its disclosure. First, the vulnerability information published by SecurityFocus is from the discussion on the Bugtraq mailing list and vendor statements, while CERT researchers perform additional research, which is eventually published in its "vulnerability notes". Second, CERT notifies vendors before publishing information and provides them with a protected period, typically 45 days. Though an identifier of a vulnerability may also notify the vendor before posting the vulnerability information on SecurityFocus, SecurityFocus generally does not contact the vendor itself. The default in SecurityFocus is *instant disclosure*. Third, CERT 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 is more selective about the vulnerabilities it publishes, publishes information that has undergone greater pre-publication scrutiny, communicates with vendors regularly, and provides them with a longer protected period.

### **4.3 Sample generation**

From 9/26/2000 to 8/11/2003, the NVD documents a total of 493 vulnerability notes published by CERT. We have included all of them in our dataset. During the same time period, the NVD documented 2681 vulnerabilities published by SecurityFocus of which 354 vulnerabilities were also published by CERT. From the 2314 SecurityFocus published vulnerabilities that are not published by CERT, we randomly sampled 131 vulnerabilities. To ensure a representative sample of vulnerability from each year of this time period, we use a stratified sampling technique, where the strata are the years in which the vulnerabilities were disclosed.

For some of the vulnerabilities published by both CERT and SecurityFocus, discrepancies were found between the reports published by SecurityFocus and CERT. For example, CERT tends to have a longer list of vulnerable vendors. For the empirical analysis, we have included all the vendors that are listed as vulnerable by CERT. When the discrepancy is about the date a vulnerability is known to be public, we have used the earlier date as the date of disclosure.

In all, this resulted in a sample of 624 vulnerabilities: 354 vulnerabilities are published by both SecurityFocus and CERT, 139 are published by CERT alone, and 131 are published by

SecurityFocus alone. A total of 643 vendors are affected by these vulnerabilities, resulting in 4874 observations. Of these, 2944 observations involved vendors that were listed as “not vulnerable” or “vulnerable status unknown” by CERT. We dropped these observations and retain only those vendors listed as *vulnerable* by CERT. This resulted in 596 vulnerabilities and 1930 observations. From this, we dropped observations where the vendor notification date or the patch release date cannot be determined (see section 5.2 for more details). Further investigation revealed that 18 vulnerabilities had been disclosed (by other sources) before year 2000, although they were officially published by CERT and/or SecurityFocus after 9/26/2000. We dropped these vulnerabilities from our sample as well. Our final sample contains 420 distinct vulnerabilities. A total of 310 vendors are affected by these vulnerabilities. These constitute 1429 observations, of which 158 observations come from SecurityFocus alone involving 57 vendors and 89 vulnerabilities (i.e., CERT did not publish them at all), 99 observations come from CERT alone involving 52 vulnerabilities and 37 vendors (i.e., SecurityFocus did not publish them all), while the rest were jointly published by both.

#### **4.4 Definition of Important terms**

Vendor Notification - Vendor notification is defined as the time that a vendor is notified of the vulnerability. CERT provided information to us (as well as in its vulnerability notes) about when it contacted the vendor. For SecurityFocus, if the identifier contacted the vendor, the vendor notification date is available. In the absence of advance notification, vendor notification is assumed to take place when the vulnerability is first made public by sources other than the vendor itself.<sup>11</sup> Therefore, in such cases, disclosure and vendor notification are assumed to take place on the same day.

Vulnerability Publication: The date when CERT or SecurityFocus publishes information about a vulnerability on its website. Note that publication is only one of ways in which a vulnerability is disclosed. Sometimes, some third parties (including some other vendors) disclose information on other forums, which is later published by CERT or SecurityFocus.

---

<sup>11</sup> When the vulnerability is first made public by a vendor itself, the true notification date and therefore the patch release time cannot be determined. As noted earlier, these observations are excluded from the final sample.

Patch Release Time: The patch release time is our dependent variable, measured as the number of days elapsed between vendor notification date and the patch release date. The information on whether and when a vendor released a patch is obtained from the vendor's response to CERT notification for vulnerabilities published by CERT. For a vulnerability published by SecurityFocus alone we determined the patch release date through a variety of sources, including the vendor's announcement of advisory regarding the vulnerability, release notes of the patch, and time-stamps of the patch. Though we have exercised great care in collecting information on patch release times, measurement error is possible. However, measurement error in the dependent variable does not introduce bias in the estimation (Green 1992).

Vulnerability Disclosure: We define disclosure as the event when information about the vulnerability becomes public. Disclosure can happen when CERT or SecurityFocus publish information about the vulnerability. However, the vulnerability information can be made public by other parties as well. The sources of disclosure include SecurityFocus, CERT, and third parties, like researchers or other vendors. CERT vulnerability notes provide information on the date the vulnerability was known to be public.

In our paper, disclosure is defined as a dummy variable such that disclosure of vulnerability  $i$  for vendor  $j$  at time  $t$  takes value 1 if vulnerability  $i$  is disclosed before time  $t$  and vendor  $j$  has not released a patch for it, and zero otherwise. Whenever vulnerability information is made public by anyone before the patch has been released, the vulnerability is disclosed. Note that, by definition, *disclosure* is a time varying variable. For a vulnerability  $i$  and vendor  $j$ , if the vendor is notified at  $t_0$  and disclosure happens at time  $t_1$  such that  $t_0 < t_1 < t$  then, *disclosure* = 0 before  $t_1$  and *disclosure* = 1 after  $t_1$ . There are two extreme cases as well. The observations with disclosure always equal to 1 (disclosure happens at the same time as the vendor is notified) are referred to as *instant disclosure*. Observations with disclosure always equal to 0 (vulnerability is made public at the same time as the vendor release a patch) are referred to as *no disclosure*. Note that by definition, *no disclosure* observations are always patched.

For the same vulnerability, disclosure conditions may vary across vendors. For instance, a vendor may have released the patch ten days after notification and posted this information on SecurityFocus, which would constitute disclosure for other vendors that are also affected by the same vulnerability but have not yet released a patch.

#### 4.5 Example of a vulnerability

Before we proceed, we provide an example of how we coded a vulnerability. Consider the vulnerability ID - CVE-2001-0803 (CERT VU#172583; SecurityFocus Bugtraq #3517). This vulnerability was originally discovered by Chris Spencer of the Internet Security Systems (ISS) X-Force, and CERT was notified. The vulnerability is - “remotely exploitable buffer overflow exists in the Common Desktop environment (CDE) subprocess control service (dtspcd). An attacker who successfully exploits this vulnerability can execute arbitrary code as root.” It is rated with a CVSS Severity Score of 10 out of 10 by the National Vulnerability Database.

The CERT report listed eight vendors to be vulnerable to this vulnerability. CERT issued formal notification on 29-Oct-01 to all the vendors that were potentially affected by this vulnerability. The vulnerability was published by SecurityFocus on 06-Nov-01, which essentially disclosed this vulnerability and CERT eventually published the vulnerability on 12-Nov-01. All vendors affected by this vulnerability released patches. Patch release dates and times are summarized in the following table.

**Table 1: Vendors for Vulnerability CVE-2001-0803**

<b>Vendor</b>	<b>Patch release date</b>	<b>Patch release time (days)</b>
Compaq Computer Corporation	09-Nov-01	11
Hewlett-Packard Company	07-Nov-01	9
IBM	30-Oct-01	1
SGI	07-Nov-01	9
Sun Microsystems Inc.	12-Nov-01	14
The Open Group	10-Nov-01	12
The SCO Group (SCO UnixWare)	06-Nov-01	8
Xi Graphics	09-Nov-01	11
Notification: 29-Oct-01; Disclosure: 06-Nov-01; CERT Publication 12-Nov-01		

In this example, disclosure is 0 for IBM. For the rest, it is 0 before 06-Nov-01, and 1 thereafter.

#### 4.6 Descriptive statistics

##### **Patch release time and disclosure:**

Statistics on patch release times and disclosure for the final sample are presented in Table 2. It is important to note that a significant number of vulnerabilities are instantly disclosed in our

sample. This may appear surprising but many vulnerabilities are publicly announced first, picked up by CERT later, and patched by the vendors later.

**Table 2: statistics on patch release time and disclosure (N = 1429)**

<b>Patch Release Time (in days)</b>	
Mean	51.40 (95.47)
Median	18
% patched by end of sample period	90%
<b>Disclosure time (in days)(of the observations facing disclosure)</b>	
Mean	8.65 (30.54)
% no disclosure	14.91%
% instant disclosure	66.13%

Standard errors are in parentheses

### Severity:

We use the CVSS Score published by the NVD as our measure of vulnerability severity, which is a number between 0 and 10.<sup>12</sup> For example, the vulnerability highlighted in the example above has a severity score of 10. The CVSS Score reflects several factors, including the ease of exploitation of the vulnerability, the estimated number of affected systems, and the impact of the vulnerability.<sup>13</sup> In particular, the availability of an exploit tool or exploit code, the ability to exploit the vulnerability remotely, the number and importance of systems affected by the vulnerability, and the level of control gained by a successful exploit - all imply a higher severity metric. Table 3 summarizes the severity of the vulnerability measured by CVSS Score and the number of affected vendors per vulnerability.

**Table 3: Vulnerability severity metric (N = 420)**

	<i>Mean</i>	<i>Std. Dev.</i>	<i>Median</i>	<i>Min</i>	<i>Max</i>
<i>Severity</i>	5.96	2.42	7	1.9	10
<i>Number of affected vendors/vulnerability</i>	3.55	7.69	1	1	93

### Publication source:

<sup>12</sup> CERT also has its own severity rating. Our results are similar using either metric, but the CERT metric is available for only a subset of vulnerabilities.

<sup>13</sup> Detailed information the CVSS score can be found at <http://www.first.org/cvss/> and <http://nvd.nist.gov/cvss>.

We classified the vulnerabilities in our dataset into three groups: (i) vulnerabilities published by both CERT and SecurityFocus, (ii) published by CERT only (but not published by SecurityFocus), (iii) published by SecurityFocus only (but not by CERT). The statistics of the three groups are presented in table 4.

**Table 4: Statistics conditioned on source of publication**

	Published by CERT alone	Published by SecurityFocus alone	Published by Both
% patched	81.82%	62.66%	94.54%
% no disclosure	5.05%	15.19%	15.70%
Disclosure time (of the observations facing disclosure)	2.55 (12.66)	3.35 (13.46)	9.95 (33.16)
Severity	5.85 (2.23)	5.59 (2.16)	6.10 (2.52)
Obs / vuls	99 / 52	158 / 89	1172 / 279
Patch release time (of the patched observations)	46.05 (78.91)	70.88 (122.52)	50.05 (93.70)

Standard errors are in parentheses

As seen from the first row, a vulnerability is patched with higher probability if it is published by both sources. This probably captures the fact that more important vulnerabilities are published by both sources.

While SecurityFocus is well known for its instant disclosure policy, not all vulnerabilities published by SecurityFocus are instantly disclosed, as can be seen from the disclosure time in Table 4. The reason is, as we have discussed above, that some of the identifiers may choose to notify the affected vendors before they report the vulnerability to SecurityFocus. On the other hand, while CERT has a stated 45 days disclosure policy, in many instances the vulnerabilities get disclosed much earlier: (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 when the vulnerability is publicly disclosed, (ii) CERT may disclose the vulnerability before the 45 days period if more than 80% of the vulnerable vendors have the patch ready.<sup>14</sup> (iii) while CERT tries to keep the vulnerability confidential during the protected period, a vulnerability can be disclosed by others, such as SecurityFocus or a vendor.

### Disclosure Source:

<sup>14</sup> Conversations with CERT staff suggest this possibility.



As discussed, we distinguish between vulnerability disclosure and vulnerability publication. In the case of vulnerability CVE-2001-0803 discussed above, the disclosure was done by SecurityFocus. The key difference is that the disclosure source is unique, and refers to the entity that first makes the vulnerability public. By contrast, a vulnerability may be published by more than one entity. As noted, publication source for our sample is confined to CERT and SecurityFocus.

In Table 5 we include only those vulnerabilities that are published by both SecurityFocus and CERT. As reported in Table 4, this constitutes 279 vulnerabilities and 1220 observations. Table 5 provides key descriptive statistics by the source of disclosure for this subsample.

**Table 5: Statistics conditioned on disclosure source  
(Vulnerabilities published by both SecurityFocus and CERT)**

	CERT	SecurityFocus	Others	No disclosure
<b>Patched and unpatched observations</b>				
% patched	97.06%	90.89%	94.44%	100%
Disclosure time (of the observations facing disclosure)	20.40 (44.26)	7.02 (28.41)	2.21 (19.20)	N/A
Severity	6.88 (2.83)	5.95 (2.55)	6.12 (2.34)	6.40 (2.63)
Obs / vuls <sup>15</sup>	281 / 28	527 / 183	180 / 52	184 / 48
<b>Patched observations only</b>				
Patch release time	54.93 (95.69)	49.34 (100.07)	61.00 (110.59)	34.46 (39.47)

Standard errors are in parenthesis.

### Vendor Characteristics:

We include vendor size (measured as number of employees), and a dummy variable for whether the vendor is an open source vendor. The total number of vulnerable vendors in our sample is 310. However, we were able to find reliable information for only 142 of them. The others are mostly small or foreign vendors and we code them 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.<sup>16</sup> The reported statistics in Table 6 are averaged over only those vendors for which information is available.

<sup>15</sup> The sum of vulnerabilities reported is 324, which is greater than 294, the number of vulnerability published by both SecurityFocus and CERT. The explanation for the apparent discrepancy is that disclosure conditions, including disclosure source, may vary for the same vulnerability across vendors. As a result, the same vulnerability may feature more than once in table 5.

<sup>16</sup> Alternative specifications in which vendors with missing size were given a value of zero yield similar results.

**Table 6: Vendor Characteristics ( $N = 142$ )**

	Mean	Std. Dev.
<i>Vendor employee size (in 000's)</i>	17.60	66.12
<i>Open source</i>	0.23	0.42

**Year of publication:**

Finally, the number of vulnerabilities published in each year is summarized in the table below:

**Table 7: Number of vulnerabilities published in each year**

Number of vulnerabilities	First known to be public
64	2000
170	2001
123	2002
63	2003

We also include a dummy for operating systems related vulnerabilities and a dummy for server applications to account for differences across products and applications which may affect patch release times.

**5. Empirical Strategy**

Our goal is to examine the impact of exogenous factors, including disclosure at time  $t$ , on the patch release time. Note that running a linear regression of patch release 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 estimates how disclosure affects the probability of the vendor releasing a patch, given that the vendor has not released patch till then. Conceptually this is similar to comparing the patch release time of two vendors who have not patched until time  $t$  but one of them faces disclosure at time  $t$  while the other does 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 patch release for vulnerability  $i$  at time. Then the proportional hazards model (PHM) is

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

Where  $\lambda_0(t)$  is the baseline hazard rate at time  $t$ ;  $\beta$  is the vector of coefficients 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:

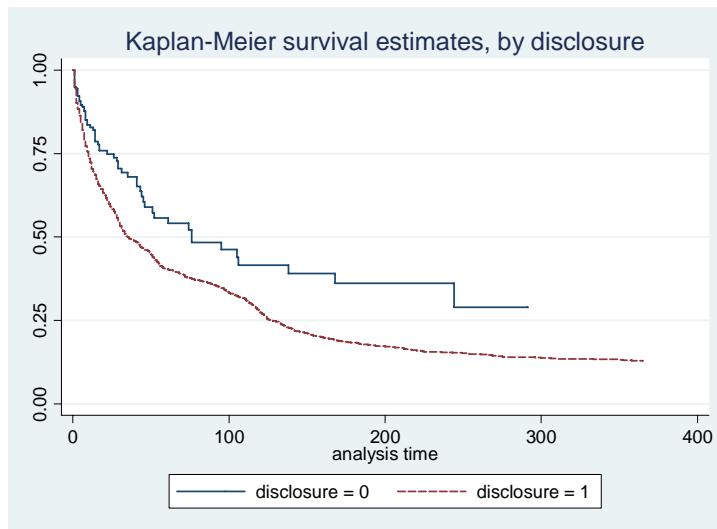
$$\lambda_{ij}(t, X, \beta) = \lambda_0(t) \exp(\beta X + e_j) \quad (2)$$

where  $i$  indexes vulnerability and  $j$  indexes vendor and  $e_j = \log(z_j)$ . The term  $z$  is unobserved frailty and is assumed to be gamma distributed with mean 1 and variance  $\sigma^2$  where the variance is to be estimated. The frailty model is akin to a “random-effects” model where a random variable  $z$  is fixed within the group of interest but varies across different groups according to a distribution. To control for vendor specific unobserved, we let  $z$  remain fixed within vendors. The idea is that the vendors may have specific patch release policies which affect their patch release decisions but are unobserved by us.<sup>17</sup>

Before proceeding to the estimations, in Figure 1 we present the Kaplan-Meier (1958) estimate of the survivor function or the probability of survival (no patch) past time  $t$ , with and without disclosure. It is clear from the figure the patch release probability is higher for disclosed vulnerabilities. By the 50<sup>th</sup> day after notification, almost 75 percent of the vulnerabilities are patched if disclosure has taken place and less than 50 percent are patched if no disclosure has taken place.

---

<sup>17</sup> We also estimated a specification where  $z$  was fixed within vulnerabilities instead (recall that a vulnerability affects multiple vendors) with results that are similar to those reported here.



**Figure 1: Kaplan-Meier Survival estimates, by disclosure**

## 5.1 Results and Discussion

There are two ways to estimate the models in (1) and (2). We can either specify the baseline hazard function ( $\lambda_0$ ) parametrically (e.g., as a Weibull distribution) or characterize it non-parametrically (as in the Cox proportional model). Parametric hazard forms are easier to work with, especially for prediction. 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 distribution is a flexible distribution with two parameters,  $\gamma$  and  $p$  and has a baseline hazard of the form  $\lambda_0 = \gamma p t^{p-1}$ . Parameter  $\gamma$  is the scale parameter and  $p$  is the shape parameter which determines the shape of the 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 8. We report hazard ratios, which are easier to interpret (the hazard ratio is simply  $\exp(\beta)$ ). A hazard ratio less than one signifies that the covariate decreases the instantaneous probability (hazard) of releasing the patch while a hazard ratio greater than one implies that the covariate increases the instantaneous probability of releasing the patch.

**Table 8: Determinants of patch release: Estimates (Hazard Ratio) Weibull Hazard Model (N = 1429)**

	(1) With frailty		(2) Without frailty	
	Haz. Ratio	Std. Error	Haz. ratio	Std. Error
<i>Disclosure</i>	2.45***	0.20	2.33***	0.18
<i>Published by CERT only</i>	0.69***	0.08	0.70***	0.08
<i>Published by SecurityFocus only</i>	0.29***	0.03	0.29***	0.03
<i>Vendor Size (Small)</i>	0.87	0.50	0.82*	0.09
<i>(1-Small)*Ln(vendor size)</i>	1.04	0.02	1.02**	0.01
<i>Open source software</i>	1.53**	0.26	1.37***	0.11
<i>Ln(Severity)</i>	1.23***	0.08	1.15**	0.07
<i>Operating System</i>	1.29***	0.10	1.31***	0.09
<i>Server Application</i>	1.25***	0.09	1.22***	0.09
<i>Published in 2001</i>	1.52***	0.19	1.58***	0.19
<i>Published in 2002</i>	2.84***	0.37	2.97***	0.37
<i>Published in 2003</i>	3.15***	0.42	3.31***	0.42
<i>Ln(p)( shape parameter)</i>	-0.55***	0.02	-0.61***	0.02
<i><math>\sigma</math> (frailty parameter)</i>	0.30***	0.08		
<i>Log Likelihood</i>	-2850		-2873	

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

The frailty parameter ( $\sigma$ ) 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 distribution is significant as well. Since  $p$  is less than 1, this suggests that the instantaneous probability of patch release (given that the vulnerability remains unpatched till then) falls over time (see Figure 1 as well).

The key variable of interest is disclosure. The estimated impact of disclosure is large and in expected direction. The instantaneous probability of patch release is almost 2.5 times higher after the vulnerability is disclosed than before. Thus hypothesis  $H1$  is supported. The estimated dummies for CERT alone and SecurityFocus alone are less than one. Since the left out category is the vulnerabilities published by both CERT and SecurityFocus, this suggests that vulnerabilities published by both sources lead to a quicker patch. This indicates that there are

important unobserved differences across vulnerabilities and that the vulnerabilities published by both sources tend to be more important.

Severity is highly significant and in the expected direction, supporting *H3*. A unit increase in the log of severity increases the instantaneous probability of patch release by about 23 percent. Another interpretation of these estimates is their impact on duration of patch release time instead of on the instantaneous probability of patch release.<sup>18</sup> Thus, we can state that a 10 percent increase in the severity score leads to about 3.6 percent decrease in the patch release time. Open source vendors also patch significantly faster than closed source vendors. On average, at any given time, open source vendors have an instantaneous probability of releasing a patch that is 1.5 times higher than the closed source vendors, which translates to about 50 percent reduction in average patch release time. Thus hypothesis *H5* is supported. Vendor size is insignificant both statistically and economically, thus hypothesis *H2*, that large vendors release patches more quickly, is rejected. Vulnerabilities in operating systems and in server applications are patched faster. Vulnerabilities that are published in later years (2003 as opposed to 2000) are patched faster. This may reflect the growing awareness amongst the consumers regarding vulnerabilities and in turn, more responsive vendors.

We also examine if the impact of disclosure varies with severity or firm size. To do this, we split the sample in low severity (below median) and high severity (above median) vulnerabilities and see if impact of disclosure is different across the sample. We do the same with firm size. We report these results in Appendix B and find that the impact of disclosure is higher for more severe vulnerabilities. We find no such evidence for the firm size.

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 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 patch release time with and without disclosure using the estimates of disclosure in Table 8. Since disclosure is a time varying

---

<sup>18</sup> These models are known as Accelerated Failure Time models (AFTs). Weibull model allows for both AFT as well as Proportional hazard model. In AFT, the dependent variable is the log of duration (patch release time in our case). The estimates from PHP model is - estimates on  $AFT * \rho$  where  $\rho$  is the estimated variance of the error term (estimate on  $\rho$  in 0.57 in our model). Thus PHP estimate of 1.23 on severity can be restated as AFT estimate of  $-\ln(1.23) / 0.57 = -0.36$

covariate, we calculate these predictions for different disclosure times. The calculations and the formula are derived in appendix B. The results showing the impact of disclosure at various times on average patching speed are shown in Table 9.

**Table 9: The Estimated Effect of Disclosure on time to release a patch (in days)**

Disclosure time $T$	Expected patch release time if disclosed at time $T$ (1)	Expected patch release time without disclosure (2)	Effect of disclosure (2) – (1)
0 (Instant disclosure)	28	63	35
1	29	63	34
2	31	63	32
3	32	63	31
4	33	63	30
5	34	63	29
6	35	63	28
7	36	63	27
8	37	63	26
9	37	63	26
10	38	63	25

These calculations are done setting the covariates at the following values, the vulnerable vendor has the average employee size of our sample; the vulnerability is a server application and is published in 2003 by both CERT and SecurityFocus and has the average severity metric of our sample.

Thus, for a typical vulnerability, if disclosure does not take place before a patch is released, on average, the patch will come in about 63 days. However, if the vulnerability has been disclosed immediately then patch would come in approximately 28 days. Thus, instant disclosure will hasten the patch by almost 35 days. Of course, this does not mean that instant disclosure is the optimal policy. Recall that disclosure exposes users to much higher risks of attacks till the patch arrives. With instant disclosure, users are exposed for 28 days. Delayed disclosure also provides the vendor with time to develop and test the patch. Without additional data on the losses from exposure and the time-cost tradeoffs in patch development, the optimal disclosure point cannot be computed. 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 times can be understood and measured. This is the key contribution of our paper.

To test hypothesis  $H4$ , we estimate the model using the samples that are published by both CERT and SecurityFocus with the disclosure source included as well. Recall that in our data, a vulnerability could be first disclosed by CERT, SecurityFocus or a third party. We include three separate dummies to indicate disclosure by these sources. As before, these dummies are time varying. The reference category is all observations with *disclosure* equal to 0. We present the results with and without frailty in Table 10.

**Table 10: Estimates (Hazard Ratio) with Disclosure source included (N = 1218)**

	(1) With frailty		(2) Without frailty	
	Haz. Ratio	Std. Error	Haz. Ratio	Std. Error
<i>Disclosed by CERT</i>	3.91***	0.42	3.38***	0.34
<i>Disclosed by SecurityFocus</i>	2.34***	0.23	2.25***	0.20
<i>Disclosed by Others</i>	2.58***	0.31	2.58***	0.29
<i>Vendor Size (Small)</i>	0.67	0.38	0.86	0.11
<i>(1-Small)*Ln(size)</i>	1.02	0.03	1.03**	0.01
<i>Open source software</i>	1.43**	0.26	1.40***	0.12
<i>Ln(Severity)</i>	1.20**	0.09	1.12*	0.08
<i>Operating System</i>	1.21**	0.10	1.27***	0.10
<i>Server Application</i>	1.36***	0.11	1.32***	0.11
<i>Published in 2001</i>	1.79***	0.26	1.85***	0.26
<i>Published in 2002</i>	2.92***	0.44	3.08***	0.45
<i>Published in 2003</i>	3.63***	0.56	3.76***	0.55
<i>Ln(p) (shape parameter)</i>	-0.51***	0.02	-0.57***	0.02
<i><math>\sigma</math> (frailty parameter)</i>	0.26***	0.08		
<i>Log Likelihood</i>	-2306		-2324	

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

We find that, all else the same, vulnerabilities initially disclosed by CERT have a higher instantaneous probability of releasing a patch. The difference ( $3.91 - 2.58 = 1.33$ ) captures the true impact of disclosure by CERT over other sources, netting out the effect of unobserved differences across vulnerabilities. Put differently, the difference in patch release times between vulnerabilities disclosed by CERT and by other sources potentially points to CERT's credibility with vendors, their research into the vulnerability and CERT's ability to communicate with the right persons inside the vendor organization. Since we identify this effect using the sample of



vulnerabilities that are published by both sources, we eliminate possible selection issues. Other estimates are also consistent with results in Table 9, indicating the robustness of our analysis.

## 5.2 External Validity

Brian Krebs of the Washington Post posted some data on Microsoft’s patch release times for some vulnerabilities on his web site ([http://blog.washingtonpost.com/securityfix/2006/01/a\\_time\\_to\\_patch.html](http://blog.washingtonpost.com/securityfix/2006/01/a_time_to_patch.html)). This additional dataset provides an opportunity to verify our results. The data covered 3 years (2003, 2004 and 2005) and Krebs collected the data using public sources as well as directly from Microsoft staff. Krebs also collected data on when Microsoft became aware of each vulnerability (which roughly corresponds to notification in our terms.) Since our data are hand collected and do not extend beyond 2003, validating our results using third party data for a later period is useful. Moreover since these vulnerabilities were first published by Microsoft (the vendor), it adds richness to our sample from the previous section.

The dataset collected by Krebs contains 99 vulnerabilities, of which 83 were first disclosed by Microsoft itself, at the time when it released patches for the vulnerabilities in question. The average severity score is 7.06 and the mean patch release time is 130 days. The remaining 16 vulnerabilities were disclosed before Microsoft could release a patch; 14 were disclosed instantaneously and 2 were disclosed after Microsoft became aware of them. The average severity score is 6.4 with mean patch release time of 62 days for the disclosed vulnerabilities. There are no other covariates because all observations come from the same vendor.<sup>19</sup> We estimate a Cox model (a Weibull model leads to similar estimates).

**Table 11: Estimates with Microsoft Data (Krebs’s Sample, N = 99)**

	Haz. ratio	Std. Error
<i>Disclosure</i>	3.05***	0.81
<i>Ln(Severity)</i>	1.53***	0.44
<i>Log Likelihood</i>	-352.04	

\*\*\* indicates significance at 1% level.

<sup>19</sup> CVE stopped including some vulnerability characteristics for the data in later periods (2003 onwards). Therefore, we cannot ascertain easily if the vulnerability pertained to operating systems or server applications and hence do not include those dummies.

As is evident from Table 11, the impact of disclosure in Krebs' sample is large and significant and even greater than the impact estimated in our sample. Severity is also significant and in expected direction. More severe vulnerabilities lead to a quicker patch. Both of these results are reassuring. Interestingly, Krebs also shows, in a separate sample (which we could not match with the NVD dataset and hence do not use) that an open source vendor like Mozilla is quicker to patch than Microsoft. This also confirms our estimate on open source.

### **5.3 Robustness of the estimates**

We also perform various other robustness checks. Note that our results are not sensitive to the assumption of the Weibull distribution since the Cox non-parametric specification yields similar results (see Appendix D). 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 patch release times by including only patched observations or dropping patch release times of more than 1 year. The impact of disclosure remains qualitatively unchanged. In short, we find that the disclosure effect is quite prominent and significant and unlikely to be the artifact of modeling technique or data collection strategies. We also tested the assumption of proportionality (which is required in a PHM hazard specification). While we detected some evidence of non-proportionality in CERT estimate, after correcting it, the results do not change in any significant fashion.

#### Endogeneity of Disclosure

One of issues is whether disclosure is exogenous in our model. For example, are there some systematic variations in vulnerability characteristics that make their disclosure faster and affect the patch release time as well? One big factor would be severity. It is possible that severe vulnerabilities are disclosed earlier and patched faster. However, we control for severity. To detect differences between instantly disclosed and non-instantly disclosed vulnerabilities, we present key statistics in table 12. In general, they do not seem to be very different (if anything, the severity of non-instantly disclosed vulnerabilities is higher). Vendor size is the only difference across the two, which again is explicitly controlled for in our estimation.

**Table 12: Comparison between Instantly and Non-instantly-disclosed vulnerabilities**

	Non-Instant- Disclosure	Instant Disclosure
<i>Obs/Vul</i>	484/113	945/346
<i>CVSS Score</i>	5.90 (2.5)	6.03 (2.4)
<i>Vendor employee size (in 000's)</i>	28.81 (83.2)	17.08 (67.8)
<i>Open source</i>	23%	25%

Table 12 is reassuring because an instrumental variable approach (the only option for dealing with endogeneity of disclosure) is extremely difficult in a hazard model (see Belzil 1995, page 116) and potentially not as reliable as the commonly used two-stage least squares approach. Therefore, we do not pursue it any further.

In this paper, a vulnerability is considered disclosed if CERT published it, SecurityFocus published it, or it was already public as reported in CERT data. Usually large vendors automatically inform CERT or SecurityFocus when they release the patch or SecurityFocus picks up that information and publishes it. However, it is conceivable that every time a vendor releases a patch on its website, it constitutes disclosure for other vendors. To account for this, we estimate our model in Table 10 by adding an additional dummy “disclosed by other vendors” (even if this patch release information was not captured by CERT or SecurityFocus). The estimates are presented in Appendix E. It can be seen that this estimate is also statistically significant and leads to the same conclusion that disclosure effect and CERT effect are significant.

In our paper, a significant number of vulnerabilities are disclosed instantly. To make sure that our results are not driven by instant disclosure, we re-estimate the model but without the observations with instant disclosure (see Appendix F). The results are again consistent that disclosure effect is robust and significant.

## 7. Conclusions

In recent times, vulnerability disclosure has become a controversial topic. 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 need to decide what the

reasonable time is for vendor to come up with a patch. Critical to any such policy is how vendor patch release behavior is affected by disclosure. To our knowledge, this is the first systematic empirical examination of this question. In particular, a policy maker can use these results to decide how disclosing the vulnerability at any given time will affect a vendor's expected patch release time.

We estimate a proportional hazard model of patch release times from a unique data set that we collected. We find that disclosure accelerates the vendors' patch release time. This verifies earlier conjectures that vendors respond to disclosure. In particular we find that instant disclosure increases the instant probability of patch release by almost 2.5 times. Put differently, compared to no disclosure, instant disclosure will force a vendor to release the patch 35 days earlier. Severe vulnerabilities are patched sooner than less severe ones, consistent with a rational model in which vendors internalize some of the customers' losses. We are also able to identify the CERT effect by highlighting that the vendors are more responsive to vulnerabilities disclosed by CERT. This probably reflects the stronger lines of communication between CERT and vendors, the value of the vulnerability analysis by CERT, and hence the reputation of CERT. Open source vendors are quicker to release a patch than the closed source vendors when vulnerability is found in their products. Given the attention on the open source versus 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). We also tested our model on a different data set for external validity and find that our results on disclosure are consistent.

While our results are interesting, there are qualifications. In particular, our model doesn't control for the patch quality. 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 threat of disclosure. Future work which can distinguish the impact of actual disclosure from threat of disclosure will be very useful for policy makers and vendors. Our model is also reduced form in nature. Future work can estimate more precise structural models. Though 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 attack propensity changes with disclosure, and the resulting changes in the extent of customer losses. We also need the estimates of the time-cost tradeoffs for vendors in

patch development. 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., T. Moore (2006), "The Economics of Information Security", *Science*, 27 (314), 610-613
- Anderson R. (2001), "Why Information Security is Hard," *In Proceedings of the 17<sup>th</sup> 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, J. McHugh (2000), "Windows of vulnerability: A case study analysis," *IEEE Computer*.
- Arora A., J.P. Caulkins, R. Telang (2006), "Sell first, Fix later: Impact of patching on software quality," *Management Science*, 52(3), 465-471.
- Arora A., R. Telang, Hao Xu (2007), "Optimal Policy for Vulnerability Disclosure", *Management Science*, Forthcoming.
- Arora A., A. Nandkumar, R. Telang (2006), "Impact of Patches and Software Vulnerability Information on Frequency of Security Attacks - An Empirical Analysis", *Information systems Frontier*, 8(5), 350-362.
- August T., T. Tunca (2005), "Network Software Security and User Incentives", *Management Science*, 52 (11), 1703-1720.
- Banker R., G. Davis, S. Slaughter (1998), "Software Development Practices, Software Complexities, and Software Maintenance", *Management Science*, 44:4, 433-450.
- Belzil C. (1995), "Unemployment Insurance and Unemployment over Time: An analysis with Event History Data", *The Review of Economics and Statistics*, 77(1), Feb 1995, 113-126.
- Camp L., C. Wolfram (2000), "Pricing Security" *In Proceedings of the CERT Information Survivability Workshop*, Boston, MA Oct. 24-26. 31-39.
- Cavusoglu H., H. Cavusoglu, S. Raghunathan (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, R. Krishnan (2005), "Software Diversity for Information Security", *The Fourth Workshop on Economics of Information Systems*. Boston, MA.
- Gordon L.A., M.P. Loeb (2002). "The Economics of Information Security Investment". *ACM Transactions on Information and System Security*, 5.

- Gordon S., 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., R.L. Prentice (2002), *The Statistical Analysis of Failure Time Data*, 2<sup>nd</sup> 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., P. Meier (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 (2007), "To disclose or not? An analysis of software user behavior" *Information Economics and Policy*, 19 (1), 43-64
- 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., M.D. Smith (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. Watal (2007), "Impact of Vulnerability Disclosure on Market Value of Software Vendors: An Empirical Analysis", *IEEE Transactions on software Engineering*, 33(8), 544-557.
- Therneau T.M., P. M. Grambsch (2000), *Modeling survival data: extending the Cox model*, New York, Springer.
- Vaupel J. W., K.G. Manton., E. Stallard (1979), "The impact of heterogeneity in individual frailty on the dynamics of mortality", *Demography*, 16, 439-454
- Wheeler D. (2002). Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! <https://wideopennews.com/archives/open-source-now-list/2002-May/pdf00000.pdf>. Accessed 19 June, 2007.

## Appendix A

The time gap between when attackers discover (or are told of) the vulnerability and when the patch is released is  $\tau - z$ , and the customer loss therefore is  $l(\tau - z)$ . It is reasonable to assume that customer losses are increasing at an increasing rate in the gap, i.e.,  $l'(\cdot) > 0$  and  $l''(\cdot) > 0$ . It follows directly that  $E_{z < \tau} l(\tau - z) < l(\tau)$ , where  $E_{z < \tau}$  represents the expectation operator conditional upon  $\tau > z$ . It is also reasonable to assume that the cost of patch development falls with the time taken to patch, albeit at a decreasing rate, i.e.,  $C'(\cdot) < 0$ ,  $C''(\cdot) > 0$ . Let  $\tau_{nd}$  be the optimal choice of  $\tau$  when there is no disclosure and  $\tau_d$  be the optimal choice when the vulnerability has been disclosed. In an interior solution,  $C'(\tau_{nd}) + \lambda E_{z < \tau} l'(\tau_{nd} - z) = 0$ , and  $C'(\tau_d) + \lambda l'(\tau_d) = 0$ .

### Hypothesis 1:

Since  $l''(\cdot) > 0$ ,  $E_{z < \tau} l'(\tau - z) < l'(\tau)$  for all  $\tau$ . It follows that  $C'(\tau_{nd}) + \lambda l'(\tau_{nd}) < 0$ . Note that  $C''(\cdot) + \lambda l''(\cdot) > 0$ . This implies that  $\tau_d < \tau_{nd}$  because by definition  $C'(\tau_d) + \lambda l'(\tau_d) = 0$ . This shows that the vendor will patch more quickly under disclosure than otherwise.

### Hypothesis 2:

An increase in size can be modeled as an upward shift in the customer loss function  $l(\cdot)$ . A simple way to accomplish this is to assume a proportionate shift. This is formally equivalent to assuming that larger firms have higher  $\lambda$  values. It is easy to see that an increase in  $\lambda$  will reduce  $\tau$  in both the disclosure and the non disclosure case.

### Hypothesis 3:

The proof follows directly upon noting that more severe vulnerabilities imply higher customer losses, and therefore, are formally equivalent to an increase in  $\lambda$ .

**Appendix B: Testing if Impact of disclosure varies with severity and firm size.**

	Frailty Model			
	High Severity (N = 669)		Low Severity (N = 760)	
	Above median		Below median	
	Haz. ratio	Std. Error	Haz. ratio	Std. Err.
<i>Disclosure</i>	3.06***	0.36	2.00***	0.24
<i>Published by CERT only</i>	0.62*	0.18	0.73**	0.10
<i>Published by SF only</i>	0.37***	0.09	0.30***	0.04
<i>Small</i>	0.57	0.38	0.96	0.45
<i>(1-Small)*Ln(size)</i>	1.00	0.03	1.05**	0.03
<i>Open source</i>	0.98	0.22	1.95***	0.38
<i>Ln(Severity)</i>	3.56***	1.68	1.21*	0.13
<i>Operating System</i>	1.29**	0.16	1.31***	0.14
<i>Server Application</i>	1.11	0.14	1.30***	0.12
<i>Published in 2001</i>	1.15	0.32	1.53***	0.23
<i>Published in 2002</i>	2.39***	0.65	2.61***	0.41
<i>Published in 2003</i>	2.70***	0.75	3.07***	0.50
<i>Ln(p) (shape parameter)</i>	-0.46***	0.03	-0.61***	0.03
<i>σ (frailty parameter)</i>	0.38***	0.11	0.17***	0.10
<i>Log Likelihood</i>	-1295.6		-1543.9	

The results indicate that the estimate on disclosure is larger for more severe vulnerabilities.

	Frailty Model			
	Large Size (N = 683)		Small Size (N = 746)	
	Above median		Below median	
	Haz. ratio	Std. Error	Haz. ratio	Haz. ratio
<i>Disclosure</i>	2.35***	0.25	2.53***	0.33
<i>Published by CERT only</i>	0.83	0.14	0.57***	0.10
<i>Published by SF only</i>	0.33***	0.06	0.27***	0.04
<i>Open source</i>	1.14	0.36	1.59**	0.35
<i>Ln(Severity)</i>	1.23**	0.12	1.18*	0.12
<i>Operating System</i>	1.19	0.13	1.47***	0.15
<i>Server Application</i>	1.49***	0.16	1.07	0.11
<i>Published in 2001</i>	1.45**	0.26	1.50**	0.28
<i>Published in 2002</i>	2.15***	0.40	3.60***	0.66
<i>Published in 2003</i>	2.77***	0.54	3.46***	0.65
<i>Ln(p) (shape parameter)</i>	-0.50***	0.03	-0.58***	0.03
<i>σ (frailty parameter)</i>	0.25***	0.10	0.49***	0.23
<i>Log Likelihood</i>	-1349.4		-1484.1	

The results indicate that the estimate on disclosure remains the same for large or small firm size.



**Appendix C: 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  $1-F(t) = f(t)/\lambda(t)$  where  $F(\cdot)$  and  $f(\cdot)$  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  $\beta$ ,  $\gamma$  and  $p$ , we can construct  $F(t | \text{disclosure} = 1, x = X)$  and  $F(t | \text{disclosure} = 0, x = X)$  for any give observation  $X$ . Ignoring  $X$  for convenience of notation, we define

$$\begin{aligned} P(t|T) &= \Pr(\text{patch before } t | \text{disclosure at } T) \\ K(t) &= \Pr(\text{patch before } t | \text{disclosure} = 0) = F(t | \text{disclosure} = 0) \\ G(t) &= \Pr(\text{patch before } t | \text{disclosure} = 1) = F(t | \text{disclosure} = 1) \end{aligned}$$

Assuming that disclosure is not affected by the decision to release a patch, it follows that when a patch is released before disclosure, the timing of disclosure is irrelevant (as long as it is after the patch is released).

$$\text{Then } P(t | T) = \begin{cases} K(t) & \text{if } t \leq T \\ (1 - K(T)) \left[ \frac{G(t) - G(T)}{1 - G(T)} \right] + K(T) & \text{if } t > T \end{cases}$$

For  $t \leq T$ , 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  $P(\cdot)$  is a well defined distribution function, because  $P(\cdot)$  is monotonically increasing in  $t$ ,  $P(0) = K(0) = 0$ ,  $P(\infty) = 1$ , and  $P(\cdot)$  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  $E(\cdot)$  for various values of disclosure time  $T$ .

**Appendix D: Estimates using non-parametric Cox model (N=1429)**

	(1)		(2)	
	With frailty		Without frailty	
	Haz. ratio	Std. Error	Haz. ratio	Std. Error
<i>Disclosure</i>	2.64 <sup>***</sup>	0.21	2.59 <sup>***</sup>	0.20
<i>Published by CERT Only</i>	0.72 <sup>***</sup>	0.09	0.73 <sup>***</sup>	0.09
<i>Published by SF Only</i>	0.35 <sup>***</sup>	0.04	0.35 <sup>***</sup>	0.04
<i>Small</i>	0.97	0.34	0.88	0.10
<i>(1-Small)*Ln(size)</i>	1.03 <sup>*</sup>	0.02	1.02 <sup>*</sup>	0.01
<i>Open source software</i>	1.45 <sup>***</sup>	0.19	1.35 <sup>***</sup>	0.11
<i>Ln(Severity)</i>	1.03 <sup>**</sup>	0.01	1.02 <sup>*</sup>	0.01
<i>Operating System</i>	1.25 <sup>***</sup>	0.09	1.26 <sup>***</sup>	0.09
<i>Server Application</i>	1.22 <sup>***</sup>	0.09	1.20 <sup>***</sup>	0.08
<i>Published in 2001</i>	1.27 <sup>*</sup>	0.16	1.28 <sup>**</sup>	0.16
<i>Published in 2002</i>	2.07 <sup>***</sup>	0.26	2.10 <sup>***</sup>	0.26
<i>Published in 2003</i>	2.25 <sup>***</sup>	0.29	2.29 <sup>***</sup>	0.29
<i>σ (frailty parameter)</i>	0.10 <sup>***</sup>	0.05		
<i>Log Likelihood</i>	-8197		-8203	

**Notes:** <sup>\*</sup> significant at 10%, <sup>\*\*</sup> significant at 5%, <sup>\*\*\*</sup> significant at 1%

**Appendix E: Estimates with Disclosure source (N = 1218)  
Including vendor patch as a disclosure source**

	(1)		(2)	
	With frailty		Without frailty	
	Haz. Ratio	Std. Error	Haz. ratio	Std. Error
<i>Disclosed by CERT</i>	4.56 <sup>***</sup>	1.09	4.59 <sup>***</sup>	1.08
<i>Disclosed by SecurityFocus</i>	2.28 <sup>***</sup>	0.37	2.28 <sup>***</sup>	0.36
<i>Disclosed by Others</i>	2.50 <sup>***</sup>	0.44	2.61 <sup>***</sup>	0.46
<i>Disclosed by Vendor</i>	2.79 <sup>***</sup>	0.45	2.83 <sup>***</sup>	0.45
<i>Small</i>	0.84	0.33	0.86	0.10
<i>(1-Small)*Ln(size)</i>	1.03	0.02	1.03 <sup>**</sup>	0.01
<i>Open source software</i>	1.46 <sup>***</sup>	0.21	1.31 <sup>***</sup>	0.11
<i>Ln(Severity)</i>	1.14 <sup>*</sup>	0.08	1.09	0.07
<i>Operating System</i>	1.29 <sup>***</sup>	0.10	1.29 <sup>***</sup>	0.10
<i>Server Application</i>	1.50 <sup>***</sup>	0.12	1.44 <sup>***</sup>	0.11
<i>Published in 2001</i>	1.63 <sup>***</sup>	0.21	1.64 <sup>***</sup>	0.21
<i>Published in 2002</i>	2.37 <sup>***</sup>	0.31	2.48 <sup>***</sup>	0.31
<i>Published in 2003</i>	3.84 <sup>***</sup>	0.52	3.84 <sup>***</sup>	0.51
<i>Ln(p) (shape parameter)</i>	-0.47 <sup>***</sup>	0.02	-0.51 <sup>***</sup>	0.02
<i>σ (frailty parameter)</i>	0.13 <sup>***</sup>	0.06		
<i>Log Likelihood</i>	-2423		-2431	

**Notes:** <sup>\*</sup> significant at 10%, <sup>\*\*</sup> significant at 5%, <sup>\*\*\*</sup> significant at 1%

**Appendix F: Estimates Without Instant disclosure observations (Weibull) (N = 484)**

	(1)		(2)	
	With frailty		Without frailty	
	Haz. ratio	Std. Error	Haz. ratio	Std. Error
<i>Disclosure</i>	3.81 <sup>***</sup>	0.42	3.26 <sup>***</sup>	0.38
<i>Published by CERT only</i>	1.01	0.32	0.84	0.25
<i>Published by SecurityFocus only</i>	0.59 <sup>**</sup>	0.12	0.54 <sup>***</sup>	0.10
<i>Small</i>	0.46	0.36	1.09	0.23
<i>(1-Small)*Ln(employment)</i>	0.99	0.04	1.04 <sup>**</sup>	0.02
<i>Open source software</i>	1.27	0.39	1.28 <sup>*</sup>	0.19
<i>Ln(Severity)</i>	1.40 <sup>***</sup>	0.17	1.24 <sup>**</sup>	0.14
<i>Operating System</i>	1.21	0.18	1.24	0.18
<i>Server Application</i>	1.49 <sup>***</sup>	0.23	1.26 <sup>*</sup>	0.18
<i>Published in 2001</i>	1.95	0.99	1.71	0.76
<i>Published in 2002</i>	2.58 <sup>**</sup>	1.29	2.76 <sup>**</sup>	1.23
<i>Published in 2003</i>	2.32 <sup>*</sup>	1.15	2.75 <sup>**</sup>	1.20
<i>Ln(p) (shape parameter)</i>	-0.45 <sup>***</sup>	0.04	-0.58 <sup>***</sup>	0.05
<i>σ (frailty parameter)</i>	0.47 <sup>***</sup>	0.13		
<i>Log Likelihood</i>	-840		-863	

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

The constant for the proportional hazard model is not identified when hazard ratio is estimated.