Diss. ETH No. 18197
TIK-Schriftenreihe Nr. 93

Security Econometrics
The Dynamics of (In)Security

A dissertation submitted to the
ETH ZURICH

for the degree of
Doctor of Science

presented by

Stefan Frei

Dipl. El. Ing. ETH
born March 28, 1968
citizen of Sissach (BL), Switzerland

accepted on the recommendation of
Prof. Dr. Bernhard Plattner, examiner
Prof. Dr. Marc Dacier, co-examiner
Gunter Ollmann, co-examiner

2009

http://www.techzoom.net/publications

ISBN: 1-4392-5409-5
ISBN-13: 9781439254097

Visit www.amazon.com to order additional copies.

Abstract

Global Internet penetration and e-commerce have grown explosively over the past years. Today, information technology has become a backbone of our industry and everyday life. We would intuitively expect such an important technology to be well-monitored and protected. However, no one would dispute that the constant discovery of new vulnerabilities drives the security risks we are constantly exposed to. As risk awareness is an essential factor in human decision making, we are in need of metrics to measure and monitor the risk exposure of our networked economy and society. Research on the economic consequences of cyber attacks has dealt primarily with microanalysis of specific events, technologies or targeted organizations. The measurement of the cumulated number of disclosed vulnerabilities over time is an interesting and often cited indicator of the increasing risk exposure. However, this measure alone is not sufficient for an analysis or understanding of the processes driving risk exposure. Accurate knowledge of the vulnerability discovery-, exploit-, disclosure-, and patch-time (the lifecycle of a vulnerability) allows one to identify different types of risk and to quantify the risk exposure and evolution thereof at global scale. A metric based on the vulnerability lifecycle is vital to better understand the security ecosystem. We build a comprehensive dataset of 30,000 vulnerabilities publicly disclosed since 1996 to reconstruct the

vulnerability lifecycle. Based on this data we analyze the risk exposure and evolution thereof from a macroeconomic perspective.

Kurzfassung

In den vergangenen Jahren erlebten wir die globale Verbreitung Internets mit einem rasanten Wachstum von E-Commerce. Die Informations-, und Kommunikationstechnologie (ICT) hat sich zu einem tragenden Element des täglichen Lebens, sowohl im Privat- wie auch im Geschäftsbereich entwickelt. Kaum jemand bezweifelt, dass die seit über einem Jahrzehnt beobachtete pausenlose Entdeckung neuer Software-Sicherheitslücken als Haupttreiber der Sicherheitsrisiken gilt denen wir ständig ausgesetzt sind. Es fehlen jedoch noch immer Metriken die auf makroökonomischer Ebene die systematische Erfassung und Überwachung solcher Risiken der vernetzten Wirtschaft und Gesellschaft erlauben. Bislang hat sich die Erforschung der wirtschaftlichen Folgen von Cyber-Attacken in erster Linie auf die Mikroanalyse von bestimmten Einzelereignissen, Technologien oder angegriffenen Organisationen beschränkt. Die Zählung der im Laufe der Zeit entdeckten Sicherheitslücken ist eine interessante und vielzitierte Grösse zur Messung der wachsenden Risikoexponierung. Allerdings ist diese Metrik alleine nicht ausreichend für eine Analyse und das Verständnis der dahinterstehenden Prozesse. Genaue Kenntnisse über den "Lebenszyklus einer Sicherheitslücke" geben Aufschluss über die Dauer und Art der Risiken denen wir auf globaler Ebene ausgesetzt sind. Darauf basierende Metriken sind von Interesse um das "Security Ecosystem" besser zu verstehen. Mit unserer

Datenbank über 27'000 Schwachstellen die bis ende 2007 entdeckt wurden rekonstruieren wir den Lebenszyklus von Schwachstellen und messen die einhergehende Riskoexponierung. Auf der Grundlage dieser Daten analysieren wir das Risiko und identifizieren wichtige Prozesse und Trends im "Security Ecosystems" aus makroökonomischer Sicht.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the last two decades we have witnessed the publication of more than 27,000 software vulnerabilities and the increased importance of information systems for our economy and society. Vulnerabilities are of significant interest when the software containing them has access to the Internet. Today it is an accepted fact that most software written inevitably suffers from design and implementation weaknesses. Continued analysis of the technical aspects of software design and development alone proved insufficient to understand the ongoing release of insecure software. Economic incentives are increasingly recognized to be a main factor contributing to todays massive exposure to insecure software [2]. Insecurity is often what economists call an "externality" - a side-effect, like environmental pollution. High fixed/low marginal costs, network effects and switching costs all tend to lead to dominant-firm markets with big first-mover advantage. So time-to-market is critical. While a platform manufacturer is building market dominance, it has to appeal to vendors of its software as well as to users, and security could

get in the way. So vendors start off with minimal protection; once they have become dominant, they add security to lock their customers in more tightly [3]. To understand the processes and limitations behind the technology-driven evolution of our economy and society, knowledge on how security information is handled at large becomes of interest. Therefore we analyze the processes of what we call the *Security Ecosytem* in Chapter 3 in detail.

Qualitatively, the security risk is proportional to both the expected losses which may be caused by an event and to the probability of this event. Greater loss and greater event likelihood result in a greater overall risk. Without knowledge of individual risk factors and probabilities, the *risk exposure time* is a viable proxy of the overall risk as the event likelihood increases monotonically with the risk exposure time.

As software users we are exposed to security risks when software vulnerabilities are discovered, publicly disclosed, or exploited. We are affected either privately as home users, as corporate users working with software centrally managed by the employer, or as the society depending on networked devices which often are not perceived as software products (e.g., *mobile phones*, *traffic management*, *electrical grid*, ..). While the mere existence of software vulnerabilities *may* lead to damage for certain individuals, at the aggregate level software vulnerabilities *will* lead to damage for the society. Likewise, a traffic accident not only leads to damage for the involved individuals, the society suffers through traffic jams, imposed delays, decreased efficiency, and health care costs.

There is a clear lack of methods to systematically evaluate and measure the risk exposure and main processes of the security ecosystem at global scale [4]. We derive metrics to analyze the evolution of the security ecosystem based on information of the timing when vulnerabilities are first *discovered*, *exploited*, *patched*, and *publicly disclosed*; these events build

the *vulnerability lifecycle*, which we introduce and discuss in Chapter 3.

We identify the following problems in this area:

- To date, there is no metric that allows for a long-term analysis of the state and evolution of the risk exposure and the processes of the security ecosystem. A risk metric always depends on the individual assessment of different risk factors and expected losses. Due to the inaccessibility, privacy, or unavailability of such data, only little is known of the risk exposure of organizations or our economy as a whole. Frequently published trend and threat reports from several security vendors and organizations focus on recent events only and cannot be compared over longer periods. Such reports are mostly based on proprietary company-owned data due to economic incentives and the business model of these organizations.

- Effective risk management requires the availability of timely, accurate, and trusted security information. Many organizations publish information on new vulnerabilities and even more organizations depend on such sources for critical security decisions. However, to date there is no analysis nor empirical data available on the quality, quantity, and timeliness of these security information sources.

- For more than a decade we have witnessed the constant discovery of new vulnerabilities and exploits. How a discoverer handles information about a new security vulnerability and how vendors of the affected software react to such discoveries depends heavily on the incentives and processes in the security ecosystem. There is an ongoing debate over the best way to responsibly handle vulnerability information and the extent of phenomena such zero-day

exploits[1]. To date there is no systematic analysis or method to distinguish or measure such phenomena in the security ecosystem.

- Aside from changing the processes of the security ecosystem that finally lead to more secure software firsthand, one way to immunize software against exploitation by vulnerabilities is the installation of a security patch. Software vendors try to match the ever increasing rate of newly discovered security vulnerabilities with the release of security patches. Security patches need to be developed and tested first, and cannot be made available instantly after the discovery of new vulnerabilities. There is a lack of empirical data to measure vendors' performance to produce patches in response to new security vulnerabilities. The availability of a patch does not protect any system until users of the software eventually implement the patch. There is also a lack of large-scale empirical data on how timely end-users implement patches.

1.2 Research Problems

In this thesis, we tackle the following research problems with regard to the security ecosystem:

- *How to measure the state and the evolution of the processes of the security ecosystem on macroeconomic scale?* The interplay of the main processes in the security ecosystem have a profound impact on the level of risk we are exposed to as a society that depends heavily on communication technology. No methodology exists today to systematically measure the dynamics of security and the state and evolution of the main processes of the security ecosystem

[1] also commonly referred to as 0-day exploits

on a macroeconomic scale. In our work we introduce the vulnerability lifecycle and develop the concept of risk exposure time as a proxy to measure the risk posed by insecure software at large scale. Using this methodology we provide aggregate indicators to better understand how (in)security processes evolve and interact in the security ecosystem.

- *What are viable security information sources for security decisions and analysis?* Trusted, accurate, and timely access to security information is a key prerequisite for any risk assessment. Since the emergence of the Internet several private and government organizations collect and publish security information on a regular basis. To date there is no empirical data available analyzing the quality, quantity, and timeliness of these security information sources. We identify viable sources for security information and analyze their role in the security ecosystem.

- *How can we measure the timeliness of security patch installations of end-users on global scale?* The delay between the availability of a security patch and the time the patch is installed by the end-user is a major contributor to the total risk exposure time. Corporate users in managed environments not only benefit from various protection mechanisms not available to ordinary home users, patching of their systems is centrally managed by their organization. Patch implementation dynamics can be measured in such environments, but the data is proprietary and not publicly available. No methodology or empirical data exists on the end-users' patch implementation dynamics on global scale. We present a method to measure patch dynamics of the most used and exposed application in the Internet, the Web browser.

1.3 Contributions

In this thesis we demonstrate that publicly available data of known security vulnerabilities provides a viable basis for assessing the security risk exposure of our economy and society. The detailed contributions are listed below. We also list the conferences where each contribution was published.

- We propose a model of the security ecosystem and define the lifecycle of a vulnerability. We propose metrics derived from the vulnerability lifecycle to measure the dynamics of insecurity and the prevalence of security ecosystem processes. The time of *discovery*, *exploit availability*, *public disclosure*, and *patch availability* are important and measurable events in the life of a vulnerability. Based entirely on publicly available data we analyze these dates for more than 27,000 vulnerabilities reported from 1996 to 2008. Results are published in the SIGCOM LSAD Workshop [5], BlackHat 2006 USA conference [6], and WEIS Workshop on Economics of Information Security 2009 [7].

- We demonstrate the existence of a competitive market of multiple security information providers. We measure the quality, quantity and timeliness of information provided by several security information providers and argue that diversity and choice for free access of security information is a prerequisite to minimize the risk exposure of security vulnerabilities to our economy and society. Results are published in the FIRST annual conference 2008 [8].

- We introduce the zero-day patch share as a new metric to analyze the number of patches a vendor is able to release at the day of the public disclosure of a new vulnerability. Using this metric we measure and compare

the security processes of different software vendors; this also provides insight in the success of the responsible disclosure process. Results are published in the BlackHat 2008 Europe conference [9].

- We propose a scalable method to passively measure Web browser patch dynamics without the need of the cooperation of the end-user. In a large scale case study we use data archived by Google's global search and Web application servers between January 2007 and June 2008 to measure Web browser's patch level of more than half of the worlds Internet population. Results are published in the DefCon 16 security conference 2008 [10] and the CRITIS'09 critical infrastructure workshop [11].

Besides these core contributions, we develop the requirements for a precise definition of the disclosure date of a vulnerability [5]. We propose the establishment of a *best used before* date for end-user software to promote security awareness for a large user base [10], [12].

1.4 Outline

This thesis is structured as follows:

- **Chapter 2** reviews related work and highlights the novel aspects of this thesis compared to previous work.
- **Chapter 3** introduces the vulnerability lifecycle and the main processes of the security ecosystem.
- **Chapter 4** describes and identifies viable sources for security information and highlights their role for a healthy security ecosystem.

- **Chapter 5** presents results of our measurements and metrics based on empirical data of the vulnerability lifecycle.
- **Chapter 6** introduces the 0-day patch share as a new metric to measure the performance of vendors' patching and security communication processes.
- **Chapter 7** introduces a method that allows unbiased large-scale measurement of the patch dynamics of the global user population.
- **Chapter 8** critically reviews the results and provides an outlook on future work.

Chapter 2

Related Work

2.1 Vulnerability Lifecycle and Ecosystem

2.1.1 Economics of Security

Our society is still in an early phase of adopting the new and seemingly endless opportunities of information technology. During the embryonic phase of innovation, before the emergence of a dominant design, the industry is characterized by high levels of experimentation among producers and customers. "*The market and the industry are in a fluid stage of development. Everyone - producers and customers - is learning as they move along.*" [13]. One of the most significant changes over the past few decades has been the rise of information technology and security as important, integral parts of everyday economic and social life [14]. While people used to think that the Internet was insecure because of lack of security features after years on providing more and more security features some started to realize that a pure technical point of view is not enough to understand the ever evolving security landscape [15]. The economics of information security, an emerging area of study, has the potential to inform security from policy and economics perspectives. Following

[14] the *security ecosystem* describes the activities of creating, preventing, dealing with, and mitigating (in)security in the use of information technology. That broad definition includes private and public activities in both legal and illegal areas. The study of the security ecosystem was initiated in a simultaneous and uncoordinated manner at different institutions around 2000. In 2000, the scientists as the Computer Emergency Response Team at Carnegie Mellon proposed an early mechanism for risk assessment. The Hierarchical Holographic Model provided the first multi-faceted evaluation tool to guide security investments using the science of risk [16]. Anderson's 2001 paper "Why information security is hard" is widely believed to be the first piece of work to *explicitly* analyze *information security* from the perspective of economics. Anderson puts forward that many of the problems in security can be explained more clearly and convincingly using the language of microeconomics: network externalities, asymmetric information, moral hazard, adverse selection, liability dumping and the tragedy of the commons [3]. Also in 2001, Gordon and Leob published a paper where the authors examined the strategic use of security information from a classical business perspective [17]. The economics of information security is *cross-disciplinary* as much as *interdisciplinary* according to Pfleeger [18]. The processes in the security ecosystem are not yet described and quantitatively evaluated at large scale. Shostack and Stewart observe in their book "The new school of information security" that until today there exist no aggregated long-term indicators or indexes to better understand how the security ecosystem functions [4].

Up to 2008 quantitative measurements of the security ecosystem typically focused on partial-analysis of individual events. Research on the economic consequences of cyber attacks has dealt primarily with microanalysis of specific events, technologies or targeted organizations [18, 19]. In 2004 Duebendorfer et al. introduced a model and methodology which allows a company to

qualitatively and quantitatively estimate possible financial losses due to partial or complete interruption of Internet connectivity [20]. Often vulnerability reports simply plot the cumulative number of disclosed vulnerabilities over time [21,22] or base their analysis on much smaller or proprietary data sets. There is a clear lack of methods to systematically evaluate or measure the risk exposure of the economy on a large, macroscopic scale [2, 15] and Greenwald et al. belief that to make real progress we must establish better experimental techniques, better metrics of security, and better models [23]. To better assess the risk exposure one has to know and understand the lifecycle of vulnerabilities and the evolution thereof.

2.1.2 Vulnerability Lifecycle

In this thesis the term *vulnerability* is used as a short form for security vulnerability. According to CVE, a vulnerability is a mistake in software that can be directly used by an attacker to gain access to a system or network [24]. Computer systems are vulnerable in the sense that they are subject to hardware failure (e.g. disk crashes, power failures, component malfunctions), software failure (bugs, logical errors, etc.), unauthorised access, deliberate attempts to disrupt operation (cracker attacks, software viruses, denial of service attacks), etc. [25]. Software defect density [26–28] has been a widely used metric to measure the quality of a program and is often used as a release criterion for a software product. Very little quantitative work has been done to characterize vulnerabilities along the same lines. The key for such analysis is most often the window of exposure, the time between the discovery of a vulnerability and the availability of a patch. To better assess the risk exposure one has to know and understand the lifecycle of vulnerabilities and the evolution thereof. In 2000 Arbaugh et al. proposed a lifecycle model for vulnerabilities and measured the number of intrusions into

systems during this lifecycle. The authors applied the model to three case studies to reveal how systems often remain vulnerable long after security fixes are available [29]. In an empirical study Arora et al. analyzed *308 vulnerabilities* in 2004 and compared the information with attacks on honeypots recorded during a period of 9 weeks to measure vendor response to vulnerability disclosure in [30]. The influence of disclosing vulnerability information on the vendors performance in releasing a patch is subject of many studies, again with only few empirical data. In 2004, Cavusoglu et al. examined how a disclosure policy affects the time for a vendor to release a patch [31] while Kannan and Telang study whether market-based mechanism for vulnerability disclosure lead to a better social outcome [32]. In 2005 Qualys compared the number of exploits available to the half-life period of critical vulnerabilities [33]. This study is based on statistical data of numerous vulnerability scans and measures the effective frequency of the application of patches by users. The disclosure date of a vulnerability is key to studies of this kind. However, the disclosure date (or release date in [34]) is defined differently among papers of different authors. Without further explanation, definitions range from 'made public to wider audience' [29], 'made public through forums or by vendor' [30], 'reported by CERT or Securityfocus' [35] or 'made public by anyone before the vendor releases a patch' in [36].

These studies were based on either a very limited number of vulnerabilities or they covered only a short period of time. Further, there is no common agreement on what is considered the *disclosure date* of a vulnerability. Some papers even relied on a very vague definition. To better understand the state and the evolution of the security ecosystem at large we need to look at more vulnerabilities over a longer period of time. In this thesis we analyze more than 27,000 vulnerabilities covering more than 10 years of evolution of the security ecosystem, and we propose a concise definition for the vulnerability disclosure date.

2.1.3 Common Vulnerability Scoring System (CVSS)

Based on the need for a standard vulnerability evaluation scheme designed to rank and evaluate risk the "The Common Vulnerability Scoring System" CVSS was created in 2005. CVSS is a vendor agnostic, open industry standard designed to convey vulnerability severity and help determine urgency and priority of response. CVSS is a joint effort involving many groups including CERT/CC, Cisco, DHS/MITRE, eBay, IBM Internet Security Systems, Microsoft, Qualys, and Symantec. CVSS is currently maintained by FIRST (Forum of Incident Response and Security Teams) [37]. CVSS solves the problem of multiple, incompatible scoring systems and is usable and understandable by anyone. The CVSS model is designed to provide the end user with an overall composite score from 0 to 10 representing the severity and risk of a vulnerability. It is derived from metrics in three distinct categories that can be quantitatively or qualitatively measured.

- *Base Metrics* contain qualities that are intrinsic to any given vulnerability that do not change over time or in different environments.

- *Temporal Metrics* contain characteristics of a vulnerability which evolve over the lifetime of vulnerability, such as the exploitability or remediation level.

- *Environmental Metrics* contain those characteristics of a vulnerability which are tied to an implementation in a specific user's environment.

The current version of CVSS (version 2) was finalized and released to the public in June 2007. Through CVSS, the security industry has made progress in creating a common language for understanding vulnerabilities and threats.

2.1.4 Disclosure Debate

In the nineteenth century Auguste Kerckhoff pointed out the wisdom of assuming that the enemy knew one's cipher system [38], which developed into a debate about the security benefit of open versus closed source systems these days. Anderson found in 2002 that whether systems are open or closed makes no difference in the long run as making it either easier, or harder, to find vulnerabilities will help attackers and defendants equally [39]. In the last years we observed a vigorous debate between software vendors and security researchers whether disclosing vulnerability information is socially desirable [15, 40, 41]. This "disclosure debate", whether or not to hide security information, is controversial, but not new: it has been an issue for locksmiths since the 19th century [42, 43]. Ozment and Schechter found that the rate by which unique vulnerabilities were disclosed for the core and unchanged FreeBSD operating system has decreased over a six-year period [44], which suggests that vulnerability disclosure can improve system security. On the other hand, Rescorla only found very weak evidence [45] for this, however his analysis is based on a limited dataset. The public pressure resulting from vulnerability disclosure also helps motivate vendors to fix bugs. The optimal disclosure policy trades off some loss from the exploitation of the vulnerability after disclosure against a delay in the release of the patch, argues Arora in [46] in 2004. In the same year Arora et al. showed that public disclosure made vendors respond with patches more quickly; attacks increased to begin with, but reported vulnerabilities declined over time [30]. Unfortunately, researchers still receive legal threats from vendors [47] seeking to stop publication of vulnerability information or "proof of concept" code demonstrating the flaw. The Electronic Frontier Foundation (EFF) discusses how security researchers can reduce their legal risk when reporting vulnerabilities in [48].

The progression of the "disclosure debate" demonstrates that we are still in an early phase of adopting to the new challenges presented by the rise of information technology. To date there exist no large scale measurements to give insight how the industry adopts their vulnerability handling processes in face of the "disclosure debate".

2.1.5 Cyber-crime

Crime has been associated with man since prehistory. Cyber-crime is defined as crimes committed on the Internet using the computer as either a tool or a targeted victim. It is very difficult to classify cyber-crimes in general into distinct groups as many crimes evolve on a daily basis [49]. When it is seen as useful and profitable, organized crime is proving as flexible and adaptable in its exploitation of opportunities provided by the spread of information technology as it is in any other field for illegal activity [50]. The lure of money is changing the computer security playing field and we must reexamine our assumptions in the face of financially motivated attackers. Prior to 2000, cyber-criminals acting alone committed the majority of cybercrimes, usually in an attempt to attain notoriety within the cyber world. However, in recent years, a shift has occurred as criminals and not just amateurs are committing cybercrimes. This is due in large part to the potentially huge financial gains that can be made from the Internet with relatively little risk [51]. In 2004 Thomas et al. highlight that fraud is likely to be as prevalent in the online environment as in the conventional environment [52]. This leads to the challenge on how to continue business with malware infected customers' computers, discussed by Ollmann in [53]. "Malware" denotes malicious software and is typically used as a catch-all term to refer to any software designed to cause damage to computer system or network, whether it's a virus, spyware, et al. Selling malware is becoming a real

business, complete with advertising, marketing, and service after the sale. The convergence of criminals with technically savvy crackers is on the way [54]. Cyber-criminals discover security vulnerabilities through their own research or simply buy the needed information in underground or *black markets* for vulnerabilities [55–57]. On the other hand, iDefense [58] and Tipping Point [59], are openly buying vulnerabilities since 2003 and 2005, respectively. Their business model is to provide vulnerability data simultaneously to their customers and to the affected vendor, so that their customers can update their defenses before anyone else. McKinney analyzes the commercialization of vulnerabilities in [60].

2.1.6 Conclusion

Information technology has become an important, integral part of everyday economic and social life, with an increased impact of security challenges for society and economy. A purely technical point of view is not enough to understand the security landscape and we lack methodologies and metrics to systematically measure key aspects of the ever evolving security ecosystem. In this thesis we analyze the lifecycle of over 27,000 vulnerabilities disclosed since 1996 to shed light on the processes of the security ecosystem.

Chapter 3

The Vulnerability Lifecycle and the Security Ecosystem

With the ongoing deployment of information technology in todays economy and society, comprehending the evolution of information security at large has become much more than the mere understanding of the underlying technologies. In the last years, people have started to realize that security failures are caused as often by bad incentives as by bad design or neglected implementation: Insecurity is often what economists call an *externality*, a side-effect of using information technology, like environmental pollution [15]. Whenever a new vulnerability is discovered, various parties with different and often conflicting motives and incentives become engaged in a complex way. These players and their interactions form what we call the *Security Ecosystem*. The security impact imposed through the interplay of the actors of the security ecosystem cannot be understood and managed unless we can better measure these risks.

The goal of this dissertation is the development of metrics that help to obtain a better understanding of the state and the evolution of todays security environment from a global perspective. Due to the inaccessibility, privacy or unavailability of data, only certain aspects of the security ecosystem can be measured from the outside. It is unlikely that cyber-criminals will ever share data about their operation and software manufacturers are reluctant to publish data about their internal vulnerability handling processes. Our method to give insight in the dynamics of the security ecosystem is the analysis of the *Lifecycle of a Vulnerability*. Thus, in the following we define the lifecycle of a vulnerability and introduce a model of the security ecosystem to describe the main players of the security landscape and their interactions. The sequence of events in the vulnerability lifecycle is used to measure the main processes governing the security ecosystem. To support the understanding of these complex processes we revisit the key elements of the "disclosure debate", look at "vulnerability markets" and analyze the motivations of vendors and cyber-criminals. Finally we show how the security ecosystem can be described and analyzed quantitatively using statistical analysis of the vulnerability lifecycle. Our methods are based entirely on publicly available data from various sources.

3.1 Lifecycle of a Vulnerability

3.1.1 What is a Vulnerability?

Creating secure software remains an elusive goal. Although this is especially true when the software has gone through several versions and includes legacy code that has not been subject to the current security processes, new software can have vulnerabilities too [61]. Even with a rigid software development process, defects are introduced that may result in severe vulnerabilities [62]. While extensive testing can isolate a large fraction of the

defects, it is typically impossible to eliminate them all. Thus, the discovery of a vulnerability is recognizing that a specific defect poses a security risk. Vulnerabilities discovered during the development and testing phase of the software will usually be fixed before the product is released. However, for more than a decade we observe an increasing number of vulnerability discoveries in products already released, as shown in Fig. 5.1. The lifecycle of a vulnerability cannot be modeled without a precise definition of the term *vulnerability*. Unfortunately, we lack such a clear definition. In the field of information security, many competing definitions of a vulnerability were proposed [63, 64]. Counting or defining vulnerabilities is a delicate undertaking that depends significantly on the parties involved, and their intent. For example, if a specific software flaw is considered *a defect*, *a feature*, or *a vulnerability* differs whether you talk to a researcher, the vendor, or different users of the software. To accurately reflect the processes in the security ecosystem we delegate the decision on what counts as a vulnerability to the Common Vulnerabilities and Exposures (CVE) consortium. CVE is a commonly accepted and widely used database of identifiers for publicly known information system vulnerabilities. CVE is a *de facto* industry standard that has achieved wide acceptance in the security industry, academia, and a number of government organizations since its launch in 1999. As of January 2008, CVE lists more than 27,000 vulnerabilities in its database. We discuss CVE in detail in Chapter 4. According to CVE, a vulnerability is a mistake in software that can be directly used by an attacker to gain access to a system or network [24]. Delegation to define a vulnerability using CVE is a step towards precise quantification, this decision is justified and explained in Def. 1 of Chapter 4. For this research, we only consider vulnerabilities $v \in V$ listed in the CVE database.

3.1.2 Vulnerability Lifecycle Events

The lifecycle of a vulnerability $v \in V$ can be divided into phases between distinctive events. Each phase reflects a specific state of the vulnerability and an associated risk exposure for the users of the software affected. To capture these phases we define the events *creation*, *discovery*, *exploit availability*, *disclosure*, *patch availability*, and *patch installation* for each vulnerability. The lifecycle of a vulnerability is shown in Fig. 3.1 where the events (as of Table 3.1) are depicted on top of the timeline.

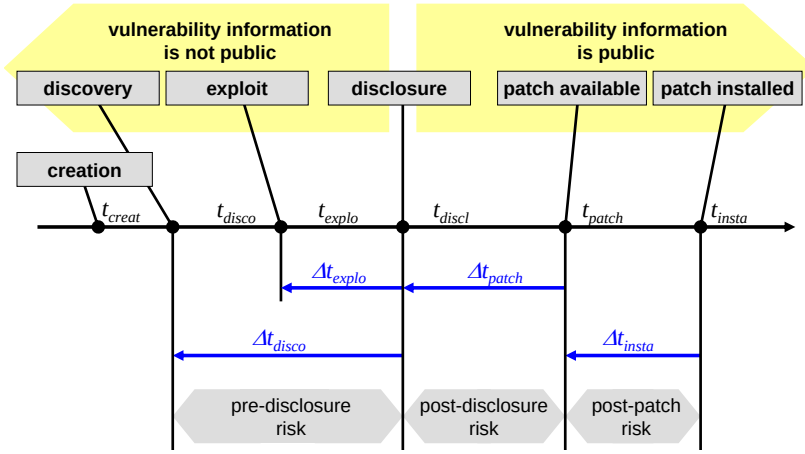

Figure 3.1: *The lifecycle of a vulnerability defined by distinctive events. The exact sequence of events varies between vulnerabilities.*

Vulnerability creation	$t_{creat}(v)$	$\rightarrow time$
Vulnerability discovery	$t_{disco}(v)$	$\rightarrow time$
Exploit availability	$t_{explo}(v)$	$\rightarrow time$
Vulnerability disclosure	$t_{discl}(v)$	$\rightarrow time$
Patch availability	$t_{patch}(v)$	$\rightarrow time$
Patch installation	$t_{insta}(v)$	$\rightarrow time$

Table 3.1: *Lifecycle times*

With some restrictions, the exact sequence of these events varies between vulnerabilities. The time of vulnerability creation $t_{creat}(v)$ and discovery $t_{disco}(v)$ are the first events in the lifecycle[1].

$$t_{creat}(v) \leq t_{disco}(v) \leq t_{\{discl,patch,insta\}}(v) \tag{3.1}$$

Further, the time of patch availability $t_{patch}(v)$ is always smaller than or equal to the time of patch installation $t_{insta}(v)$.

$$t_{patch}(v) \leq t_{insta}(v) \tag{3.2}$$

The exact sequence of these events depends on the flow of the processes explained in the security ecosystem. In the following we first discuss these events individually and relate the vulnerability lifecycle to different processes in the security ecosystem modeled in Fig. 3.2.

Time of creation (t_{creat})

Vulnerabilities are typically created by accident as the result of a coding mistake, often involving the mismanagement of memory. If a vulnerability remains undetected in the code throughout the development and testing phases, chances are it will make

[1] In rare circumstances an exploit may exist before the vulnerability has been discovered.

it into generally available code that is then released [65]. After the release of the software the vulnerability may be discovered, where discovery denotes the time the vulnerability is recognized to have a security impact. In this research we only consider vulnerabilities discovered *after* the release of the software. The time of vulnerability creation is typically unknown by definition, however it may be determined in retrospect, after the discovery or disclosure of the vulnerability. If the creation of a vulnerability is malicious and thus intentional, discovery and creation time coincide [29]. In this dissertation we do not further investigate the time of vulnerability creation.

Time of discovery (t_{disco})

The *time of discovery* is the earliest time of a software vulnerability being recognized to pose a security risk. Vulnerabilities do exist before they are discovered, but prior to the discovery of the vulnerability the underlying defect is not recognized to pose a security risk. Usually the time of discovery of a vulnerability is not publicly known until *after* its disclosure. Indeed, for many vulnerabilities the discovery time will never be known or reported to the public, depending on the motives of the discoverer.

$$t_{disco}(v) \rightarrow time \qquad v \in V \tag{3.3}$$

Time of exploit availability (t_{explo})

An exploit is a piece of software, a virus[2], a set of data, or sequence of commands that takes advantage of a vulnerability in order to cause unintended or unanticipated behavior to occur in software or an embedded device. Proof of concept code or exploits provided within security research and analysis tools are also

[2]Formally a virus is a delivery method for the exploit - not the actual exploit

deemed an exploit[3]. Typically it is a trivial exercise for criminals to turn such code into a working exploit. The *time of exploit* is the earliest time an exploit for a vulnerability is available. In rare circumstances an exploit may exist before the vulnerability has been discovered. This has happened a few times already where the exploit writer thought they were exploiting a particular (disclosed) vulnerability, only to find out later that they were exploiting an unknown vulnerability or vector - because they either didn't understand the original vulnerability or based their exploit on incorrect assumptions. Depending on the prevalence of the processes in the security ecosystem, as depicted in Fig. 3.2, an exploit is first available to cyber-criminals, security researchers, or the vendor.

$$t_{explo}(v) \rightarrow time \qquad v \in V \tag{3.4}$$

Time of public disclosure (t_{discl})

The *time of disclosure* is the first time when validated information about a specific vulnerability is made publicly available. Without further explanation, in the literature definitions of *disclosure* range from "made public to wider audience", "made public through forums or by vendor", "reported by CERT or Securityfocus", or "made public by anyone before vendor releases a patch" as in [29,30,34]. The purpose of the *disclosure* is to make the security information available to the public in a standardized, understandable format. Disclosing security information is an important event in the security ecosystem. We discuss the implications of vulnerability disclosure in the "disclosure debate" in Chapter 3 and provide a concise definition of the concept of the disclosure of a vulnerability in Chapter 4 when we discuss

[3]E.g., *Metasploit*, a tool for developing and executing exploit code to aid in penetration testing and IDS signature development.

and analyze the performance of Security Information Providers (SIP).

$$t_{discl}(v) \rightarrow time \qquad v \in V \tag{3.5}$$

Time of patch availability (t_{patch})

The *time of patch availability* is the earliest time the software manufacturer or vendor releases a patch that provides protection against the exploitation of the vulnerability[4]. Unfortunately, software vendors cannot make security patches available instantly after the discovery of new vulnerabilities or exploits. While some vendors publish patches as soon as these are available, others publish patches on a predefined schedule to ease the planning of patch installation (e.g., monthly or quarterly scheduled release of new patches). We analyze the patch release performance of various software vendors in detail in Chapter 5 and Chapter 6. In many cases a patch may be available before public disclosure (e.g., the DNS vulnerabilities of 2008 and service pack roll-ups for new operating systems [66]). Fixes and patches offered by third parties are not considered as a patch, we deem the software manufacturer as the only authoritative source to provide patches for his software. Further, enterprises often do not allow third party fixes to be installed on their systems [67]. The complexity of patches varies from simple configuration fixes to extensive changes in the foundation of the software. Other security mechanisms such as signatures for intrusion prevention systems or anti-virus tools are not considered as patches neither as these mechanisms do not eliminate the root cause of the vulnerability.

$$t_{patch}(v) \rightarrow time \qquad v \in V \tag{3.6}$$

[4]In the following of this dissertation we will use the term *vendor* to name the manufacturer of the software for *commercial products*, *freeware*, and *open-source software* alike

Time of patch installation (t_{insta})

Software users can only benefit from the immunization of a vulnerability after a patch is installed on their systems. The processes leading from patch availability to patch installation vary considerably among different kinds of software users. Hence, the time to patch installation is not a specific point in time for a vulnerability, it can only be given as a distribution for a specific sample of users. Typically business and private users face different challenges to timely patch installation. In Chapter 7 we analyze the patch installation characteristics of Internet users in detail. Installing a patch or changing security relevant configuration settings on a mission critical business system is a non trivial task for any enterprise. Further, there are business constraints that do not allow instant and automatic installation of patches on operative systems at any time. The usual way organizations approach the scheduling of patches is that they define two classes of periods throughout the year: a *patch window period* and a *freeze window period* as shown in Table 3.2 and analyzed in [67]. A patch can be scheduled for installation if the date lays in the patch window but not in the freeze window period. This general strategy is typically employed by larger organizations, which then deploy individual rules to define these periods.

Individual end-user systems are not subject to business constraints with respect to scheduling patch installation. However, in the analysis of Chapter 7 we find considerable delays of patch installation timing of end-user systems, mostly attributed to the degree of automation available to install patches.

Class	Description
Patch window	Rules to determine the periods during which the installation of patches is allowed.
Freeze window	Rules to determine the periods during which changes in the infrastructure are not allowed. Except for emergency patches, the freeze window overrides the patch window.

Table 3.2: *Classes of patch scheduling rules.*

3.1.3 Risk Exposure Times

Between the discovery of a vulnerability and its elimination through the installation of a patch, a system is potentially at risk. This exposure period can be separated into three phases: the "pre-disclosure", the "post-disclosure" and the "post-patch" phase as shown in Fig. 3.1. We analyze the relation and evolution of these periods to distinguish and understand important processes in the security ecosystem.

Pre-disclosure phase

During the time from discovery to disclosure Δt_{disco}, only a unknown group is aware of the vulnerability. This group could be anyone from lonely hackers to cyber-criminals tempted to misuse their knowledge. On the other hand, this group could also be researchers and vendors working together to provide a patch for the identified vulnerability. We call the risk exposure arising from this period as "pre-disclosure" risk because the vulnerability is known to have a security impact whereas the public has no access to this knowledge.

$$\Delta t_{disco}(v) = t_{disco}(v) - t_{discl}(v) \tag{3.7}$$

Post-disclosure phase

During the time from disclosure to patch availability Δt_{patch} the user of the software waits for the vendor to release a patch. We call the risk exposure arising from this period the "post-disclosure" risk because the public is aware of this risk but has not yet received remediation from the software vendor/originator. However, users of the vulnerable software can assess their individual risk and implement a workaround based on the information provided with the disclosure of the vulnerability.

$$\Delta t_{patch}(v) = t_{patch}(v) - t_{discl}(v) \tag{3.8}$$

Post-patch phase

The time from patch availability to patch installation Δt_{insta} is called the "post-patch" risk. The duration of this period is typically under direct control of the user of the affected software or embedded device. In organizations, this period is determined through the vulnerability management processes in place [67]. End users patch their systems manually or with the help of *auto-update mechanisms* build into their operating system or applications. In Chapter 7 we analyze and discuss the "post-patch" phase in depth. Note that an ever increasing number of embedded control devices are deployed in support of our networked society, many of which cannot be patched by its users.

$$\Delta t_{insta}(v) = t_{insta}(v) - t_{patch}(v) \tag{3.9}$$

Exogenous vs. Endogenous

We designate the "pre-disclosure" and "post-disclosure" phases as *exogenous*, since the operator of the vulnerable system cannot exert direct influence on the length of these periods. The length of these phases can only be influenced on a macro perspective

through the interplay of the processes in the security ecosystem, as shown in Fig. 3.2 and discussed in Section 3.2. Likewise, the nature of the "post-patch" phase is *endogenous* as the operator of the system determines the time when the patch is installed.

3.2 Security Ecosystem Model

In the last decade, the number of players, their roles, and interactions in the security ecosystem evolved considerably. A variety of legislative and social issues directly influence the processes of vulnerability research, detection, publication, and response. Individuals and organizations have a wide variety of motivations, some in direct conflict with each other, that add to the complexity of how to handle vulnerability information. Vendors, developers, customers, cyber-criminals, and the security community have divergent perspectives on the impact of vulnerabilities. The processes and interactions between those actors are driven by the continuous discovery of new vulnerabilities and the subsequent constant need of the public (the software users) for security information and patches. In Fig. 3.2 we model the main processes in the security ecosystem, starting with the discovery of a new vulnerability on top and the public disclosure of vulnerability information at the bottom. The flow of vulnerability information from the discoverer to the public can take several paths, each describing a different process with implications for the resulting risk exposure. The boxes *Discovery*, *Exploit*, *Patch*, and *Disclosure* in our model identify important events in the security ecosystem that can be related to events in the vulnerability lifecycle as introduced in Section 3.1. Examination of the exact sequence of vulnerability lifecycle events for a large sample of vulnerabilities allows us to identify the prevalence of particular processes and the dynamics of the security ecosystem. In the following sections of this chapter we

introduce and discuss the major players and main processes in security ecosystem. In the remaining chapters of this dissertation we present our analysis and measurements of the vulnerability lifecycle and its implications for the understanding of the security ecosystem:

Our key concept to analyze and compare the dynamics of the lifecycle of thousands of vulnerabilities is to normalize the time-differences $\Delta t_{\{diso,explo,patch\}}(v)$ as shown in Fig. 3.1 with respect to the disclosure time $t_{discl}(v)$. We first examine SIPs and introduce the information sources for this research in Chapter 4. Then we analyze the distribution of the time-differences $\Delta t_{\{disco,explo,patch\}}(v)$ based on empirical data of more than 27,000 vulnerabilities in Chapter 5. This data provides a first insight into the dynamics of the security ecosystem at large. In Chapter 6 we introduce the *0-day patch share* as a new metric to measure the patch release performance of vendors. In Chapter 7 we look at the delay between patch availability and patch implementation Δt_{insta} by analyzing the dynamics of major and minor versions upgrades of Web browsers for more than a year, covering approximately 75% of the global Internet users.

In the following sections we start with a discussion of the major players in the security ecosystem, followed with an introduction of the main processes.

3.2.1 Discoverer

The *discoverer* of a vulnerability is an individual or organization (e.g., the vendor, independent researcher, cyber-criminal, government agency) that discovers a new vulnerability. How the discoverer proceeds with this security information depends on his intrinsic motivation and the incentives offered by the environment. Whatever the choice, it ultimately impacts the risk exposure time of the public. There are many different motivations to direct the discoverer of a vulnerability:

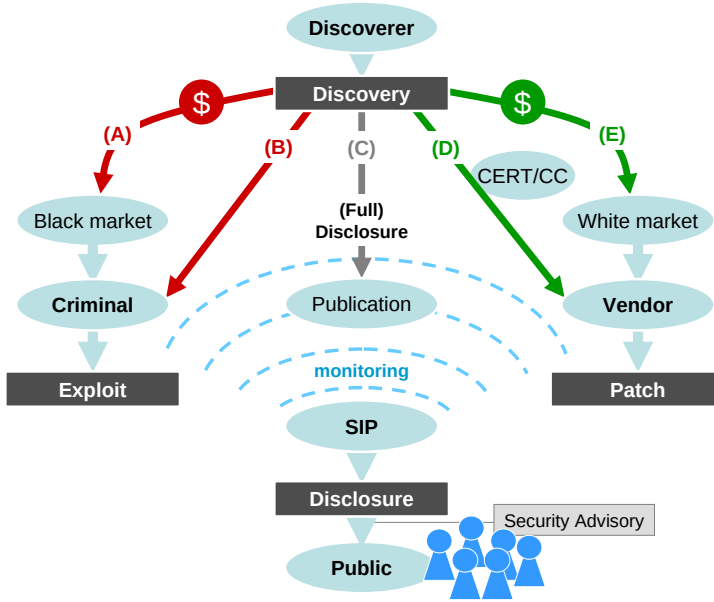

Figure 3.2: *Main processes of the security ecosystem and lifecycle.*

- altruism, make computers more secure
- recognition or fame
- self-marketing to highlight technical skills (for individuals as well as organizations)
- forcing unresponsive vendors to address a vulnerability
- curiosity and the challenge of vulnerability analysis
- malicious intent for profit making
- political motives

3.2.2 Vulnerability Markets

Vulnerability information is traded in either underground markets "black market" or as commercial services "white market". Information about security vulnerabilities can be a valuable asset. While a market for vulnerabilities has developed, vulnerability commercialization remains a hotly debated topic tied to the concept of vulnerability disclosure. Responsible disclosure fails to satisfy security researchers who expect to be financially compensated while reporting vulnerabilities to the vendor with the expectation of being paid might be viewed as extortion [48]. On the other hand, cyber-criminals not bound by legal or ethical considerations, are willing to invest considerable amounts in suitable vulnerability information. H. D. Moore[5] claims that he was offered between $60,000 and $120,000 for critical vulnerabilities in Microsoft products as reported in [60, 68, 69]. Researchers that intend to sell a vulnerability face the possibility that the same vulnerability is discovered, patched, and published independently. This threat of independent discovery pressures them to sell the vulnerability to the quickest bidder instead of the highest one. Factors that determine the market price of a vulnerability are:

- *Exclusivity of information.* This is the key factor, once the vulnerability becomes widely known the value of the information becomes zero or almost zero.
- *Security impact.* The higher the security impact, the higher the value of the vulnerability.
- *Product popularity.* A vulnerability affecting a popular product has a higher value.

[5] H. D. Moore founded the Metasploit project, an open platform for developing and testing exploit code.

To understand the role of vulnerability markets in the context of the vulnerability ecosystem, we classify vulnerability markets as black or white markets:

Black Market

The black market has developed around the illegal or malicious use of the vulnerability information. Sellers are not driven by ethical considerations and participate because they believe to make more money from black market sales than from white market sales. The trading is not openly advertised and the information is used in a way that generally increases the risk exposure of the public. The lack of trust between seller and buyers potentially exposes both parties to fraud. By the nature of this business, accurate information on the number and type of trades completed is not systematically available. Only specific investigations provide some insight into the inner workings, e.g. by Symantec's "Underground Economy Report" [70].

White Market

Players in the white market offer commercial services and openly advertise their vulnerability handling policies. Demonstrating and ensuring that buyers and sellers don't have malicious intent is a major challenge for the players in the commercial vulnerability market. White market buyers typically purchase vulnerability information to protect their customers before the vulnerability becomes public knowledge, and inform the vendor of the affected software. Such buyers advertise their ethics and ask security researchers to accept less cash (than offered in the black market) with the promise that the information will be used for benevolent purposes [60].

Incentives for the buyers are:

- Publicity generated from disclosing newsworthy vulnerabilities drives interest in their commercial services.
- Providers of intrusion detection and prevention systems include additional protection, which customers might perceive as an advantage.
- They provide the information as a payed service to their customers.

Today, the main players in the commercial vulnerability market are *iDefense* [58] which started their vulnerability contributor program (VCP) in 2003 and *TippingPoint* [59] with their zero-day initiative (ZDI) started in 2005. TippingPoint's ZDI receives an average of about 40 new vulnerabilities per month, and buys about one out of 10. Vulnerability prices are not disclosed but ZDI runs a "frequent-flier" style program that can pay out bonuses as high as $20,000 to top researchers. About 40 percent of ZDI's top researchers - the program boasts more than 600 in total - work in the security industry, according to a poll TippingPoint conducted. Just 10 percent of the researchers admitted that they would consider selling their findings to the criminal underground if they were offered more money, the poll found [71]. We discuss the prevalence of commercial vulnerability markets in Section 5.7 of Chapter 5.

3.2.3 Criminal

Any individual or organization misusing vulnerability information for its own profit for whatever motivation is denoted as *criminal* in the model of Fig. 3.2. This can be anyone from an individual hacker to cyber-criminals or government agencies. In this context *misuse* stands for any operation on the targeted system that the user of the system neither approved nor is aware of. Criminals develop or buy exploit material in order to make

use of a vulnerability and typically install malicious software to spy the user and launch further attacks.

3.2.4 Vendor

The vendor is the originator or manufacturer of the software affected by a vulnerability. We use the term vendor for commercial products, freeware, and open-source software alike. It is up to the vendor to produce and release a patch once he becomes aware of a vulnerability in his software. In Chapter 6 we introduce the 0-day patch share as a new metric to measure the performance of vendors' patching and security communication processes.

3.2.5 Security Information Provider (SIP)

In face of a fast evolving and hostile cyber-environment, businesses and private users alike are in constant need of accurate and validated security information to assess their risk and to protect their systems. There are many outlets for publicly available security information such as *vendor sites*, *security portals*, *mailing lists*, *security conferences*, and *expert blogs* to name just a few. However, for the majority of businesses and users it is unfeasible and economically prohibitive to monitor, understand and validate all the possible information sources in order to extract the security information relevant for them. Since the early days of the Internet several private and government organizations specialize in collecting and publishing security information. Some of these organizations run security research labs, sell security tools (e.g., intrusion detection systems, anti-virus software), or provide paid security and consulting services. These organizations efficiently monitor various sources of security information, validate the content found, and publish their findings as *security advisories* which describe vulnerabilities in a standardized format. These organizations have an important role in the security ecosystem

and we denominate them *Security Information Providers (SIP)*. Their constant monitoring of the (in)security environment is depicted by dashed arrows in Fig. 3.2. Through SIP services the public has a systematic access to timely, validated, and understandable security information. In Chapter 4 we show that these services provide the data needed to determine the disclosure time $t_{discl}(v)$ in the vulnerability lifecycle. Further, the availability of trusted security information from independent organizations has an important impact on the *behavior* and *incentives* on the actors in the security ecosystem:

- Businesses and private users get validated security information in a standard, understandable format allowing them to assess their risk exposure.
- Vulnerability information being published as a security advisory by an established security information provider can hardly be ignored by the affected vendor.
- The threat of publishing a vulnerability is almost as good as actually publishing it. Knowing that trusted security information providers gather, validate and disseminate security advisories directly supports the responsible disclosure process discussed in Section 3.4.

The combined effect of the efforts of independent security information providers is a major pillar building the incentives for the actors in the security ecosystem: The role of security information providers is comparable to the role of the free and independent press in an open society. Issues addressed by them can hardly be ignored, hidden or downplayed.

3.2.6 Public

The public denotes all users, individuals or organizations, that use software affected by a vulnerability. These users typically are

in need of accurate and validated security information to assess their risk and to protect their systems until a patch is released by the vendor.

3.3 Processes of the Security Ecosystem

Whether ethical or mischievous parties first get information about a new vulnerability impacts the risk exposure of software users. After the discoverer finds a new vulnerability we distinguish five principal paths (A) to (E) to proceed as depicted by solid arrows in Fig. 3.2.

3.3.1 Path (A) and Path (B)

Cyber-criminals discover security vulnerabilities through their own research or simply buy the needed information in underground or *black markets* for vulnerabilities [55–57], represented by Path (A) and Path (B) respectively. Security vulnerabilities in widely used software prove to be a formidable instrument in the hands of cyber-criminals to either enable or expand their business. Indeed, organized crime is proving as flexible and adaptable in its exploitation of cyber opportunities as it is in any other field for illegal activity when they see it as useful and profitable [50]. These paths favor the malicious use of vulnerability information resulting in an increase of security impact and exposure to risk for users. The acquirer of the vulnerability, a criminal, uses the information to build an exploit in order to misuse vulnerable systems. While the vulnerability is actively exploited, any information about it will be withheld from the public in order to delay the development of countermeasures and patches and to minimize the chance of detection. However, active use of an exploit eventually exposes the information (e.g., a sample of the exploit gets analyzed by a anti-virus lab) and the vulnerability gets reported to the vendor and the public

by SIPs. The vendor can only start developing a patch after the vulnerability is actively exploited. For a vulnerability $v \in V$ following Path (A) or Path (B) we typically observe the following sequence of events:

$$Discovery \rightarrow Exploit \rightarrow Disclosure \rightarrow Patch$$

$$t_{disco}(v) < t_{explo}(v) < t_{discl}(v) < t_{patch}(v)$$

Both paths result in a decrease of social welfare given the ubiquitous use of computer and communication technologies in our society. The time of vulnerability discovery will most likely not be found out as criminals typically do not share information.

3.3.2 Path (C)

The discoverer publishes information about the vulnerability on a suitable channel (e.g., in a security conference or on a security mailing list[6]). SIPs monitoring the security landscape spot this information and report it in a new security advisory. In the extreme case of *full disclosure* the discoverer includes proof-of-concept code and exploit material. A discoverer following Path (C) is typically not financially motivated. He either decides to publish the vulnerability firsthand, or he does so because the vendor was not responsive. We discuss full disclosure and the "disclosure debate" in detail in Section 3.4. For a vulnerability $v \in V$ following Path (C) we typically observe the following sequence of events:

$$Discovery \rightarrow Disclosure \rightarrow Exploit \rightarrow Patch$$

$$t_{disco}(v) < t_{discl}(v) < t_{explo}(v) < t_{patch}(v)$$

Following Path (C) the vulnerability information is available to all interested parties at the same time, the criminals, the

[6] FullDisclosure and BugTraq are two well known security mailing lists

vendor, and the public. Usually writing an exploit based on vulnerability information is less complex and faster than writing and releasing a patch. We present a detailed analysis of the timing between exploit and patch availability in Chapter 5. While full disclosure seems to be a poor choice with respect to the resulting security impact for the public, it is a valuable and necessary option to establish incentives in the security ecosystem that favor public security in the long term.

3.3.3 Path (D) and Path (E)

The discoverer reports the vulnerability either directly to the vendor or through a commercial vulnerability market. In case the vulnerability affects several vendors the discoverer can do so using the services of a CERT/CC[7]. The discoverer and the vendor then typically follow the "responsible disclosure" process described in Section 3.4: the vulnerability information is kept secret until the vendor has a patch ready for release. If the vendor is not responsive or cooperation fails the discoverer might revert to Path (C). When the patch is ready, the discoverer publishes his advisory at the same time as the vendor releases the patch. Criminals can only start with the development of an exploit after a patch is available. For a vulnerability $v \in V$ following Path (D) or Path (E) we typically observe the following sequence of events:

$$Discovery \rightarrow \left\{ \begin{array}{c} Disclosure \\ Patch \end{array} \right\} \rightarrow Exploit$$

$$t_{disco}(v) < t_{discl}(v) = t_{patch}(v) < t_{explo}(v)$$

Path (E) is an option for a financially motivated discoverer who does not want to sell the vulnerability in the underground where misuse is very likely. Buyers in commercial vulnerability markets typically purchase vulnerability information, inform the

[7] CERT Coordination Center

vendor of the vulnerable software, and protect/inform their own customers before the vulnerability becomes public knowledge and a patch is available. We measure the prevalence of commercial vulnerability markets Section 3.2.2 of Chapter 5. Path (D) and Path (E) are considered more favorable for the public risk exposure as the vendor gets the information about the vulnerability before mischievous parties do.

3.4 The Disclosure Debate

To support the understanding of the complex interactions in the security ecosystem we revisit the key elements of the "disclosure debate". How to report security vulnerabilities is part of a broader debate about the potential harms and benefits of publishing information that can be used for dangerous purposes. In the last years we observed a vigorous debate between software vendors and security researchers whether disclosing vulnerabilities is socially desirable [15, 40, 41]. Finding the "correct way" to handle vulnerability information still proves to be a delicate process. Appreciation of vulnerability disclosure concepts and the accompanying incentives of the players involved is a prerequisite to understand the processes in the security ecosystem. The "disclosure debate" discusses the question of how to handle information about security vulnerabilities in order to minimize the security impact for the society:

- On the one hand, public disclosure of security information enables informed consumer choice and inspires vendors to be truthful about flaws, repair vulnerabilities and build more secure products. This is the *security through transparency* stance of Kerckhoff [38].

- On the other hand, vulnerability information can give attackers (not sophisticated enough to identify a vulnerability

on their own) the very information they need to exploit a security hole in a computer or system and cause harm. This is the *security through obscurity* stance[8].

The process of *responsible disclosure* evolved as a middle-way between the opposing stances found in the "disclosure debate". It has evolved and become a widely accepted way to handle security information [34].

3.4.1 The Full Disclosure Concept

Full disclosure is a security philosophy that believes that the details of security vulnerabilities should be available to everyone in a timely fashion. The concept of full disclosure is controversial, but not new: it has been an issue for locksmiths since the 19th century [43]:

> *"A commercial, and in some respects a social doubt has been started within the last year or two, whether or not it is right to discuss so openly the security or insecurity of locks. Many well-meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty, by showing others how to be dishonest. This is a fallacy. Rogues are very keen in their profession, and know already much more than we can teach them respecting their several kinds of roguery.*
>
> *Rogues knew a good deal about lock-picking long before locksmiths discussed it among themselves, as they have lately done. If a lock, let it have been made in whatever country, or by whatever maker, is not so inviolable as it has hitherto been deemed to be, surely it is to the interest of honest persons to know this fact,*

[8] also often referred to as *bug secrecy*

> *because the dishonest are tolerably certain to apply the knowledge practically; and the spread of the knowledge is necessary to give fair play to those who might suffer by ignorance."*

Many locksmiths liked it the other way, believing that a system's security increases by keeping the general population from learning these vulnerabilities [42, 72]. This is no different than the computer world. Before the systematic publication of software vulnerabilities, vendors would not bother to spend the time and money to fix vulnerabilities, believing in the security of secrecy [42, 48, 54, 73, 74]:

Proponents of *security through obscurity* argue that publishing details of any kind of vulnerability does more harm than good by providing cyber-criminals the information needed to create tools and methods to exploit vulnerable systems. They assume that vulnerability information can be kept and controlled within a group of trusted individuals and see this as a way to protect vulnerable systems. This assumption does not hold in reality as there is no way to assure that cyber-criminals do not already possess the same vulnerability information. While vulnerability information is kept secret, benign users of the software are not aware of the risks and lack the information to defend their systems while the vulnerability is already actively exploited: a systematic information asymmetry favoring the bad. Further, the lack of public pressure implied by keeping information about vulnerabilities secret allows a vendor to delay the development of a patch, or even to deny the existence of a vulnerability.

Proponents of *full disclosure* believe that the details of security vulnerabilities should be made public as soon as possible. This means that everyone gets the same information at the same time, and can therefore analyze the impact to act upon it. Disclosure and peer review advances the state of the art in security. Researchers can figure out where new technologies

need to be developed, and the information can help policymakers understand where problems tend to occur [48]. Public disclosure or the threat of disclosure often gives vendors a strong incentive to fix the problem quickly. It is inevitable that cyber-criminals get the information alike with the public disclosure. This disadvantage is more than compensated by providing benign users the information needed to defend their systems and by the incentives created for vendors.

3.4.2 Responsible Disclosure Process

The main insights from the "disclosure debate" is that secrecy mainly prevents people from assessing their own risks which contributes to a false sense of security [40]. With the objective to minimize the security impact of vulnerabilities, the process of *responsible disclosure* evolved as a middle course between the extremes of *full disclosure* and *security through obscurity*.

The term *responsible disclosure* has come to mean that the researcher discloses full information to the vendor only, expecting that the vendor will start the process to develop a patch. In return, the vendor is supposed to expeditiously issue a patch and give credit to the researcher for his discovery. The vendor is well incentivized to collaborate and produce a patch, as the discoverer can revert to *full disclosure* at any time in case the vendor becomes unresponsive or the vulnerability is reported or discussed publicly on other channels. In the last phase of the responsible disclosure process the discoverer will coordinate the publication of his advisory with the vendor's publication of the vulnerability information and the patch. The following factors favor the responsible vulnerability disclosure process discussed:

- Well documented and published security processes, especially vulnerability handling processes

- Good track record of treating vulnerability researchers that report vulnerabilities fairly
- Referencing the discoverer of the vulnerability in the vendors' security bulleting upon the release of the patch

No or misleading documentation of security processes as well as threats against researchers reporting security vulnerabilities can at best delay the publication of vulnerability information, as does the denial or downplaying of vulnerability information reported to the vendor [47]. Not every vendor has adopted "responsible disclosure" or follows these suggestion to foster a good working relationship with the security community, as the founder of the BlackHat security conference observed in an recent interview [75]. However, an increasing number of vendors and security organizations adopted some form of *responsible disclosure* over the last decade [74, 76, 77].

3.4.3 Legal Aspects of Vulnerability Disclosure

Problems persist, although many security researchers have voluntarily adopted the delayed publication policy of *responsible disclosure*. In the past, many researchers following the *responsible disclosure* process have received legal threats from vendors seeking to stop the publication of vulnerability information. For instance, the BlackHat security conference frequently has to pull back talks in face of legal threats [75]. For example, the company NXP sued researchers from Radboud University in July 2008 to stop the publication of a paper to be released in October that explains the details on how they successfully cloned the Mifare WiFi card [47]. In [48] the Electronic Frontier Foundation (EFF) lists the following aspects of vulnerability reports that are legally most risky:

- The more detailed the advisory, the more risky it is.

- The more functional code an advisory contains, the more risky it is.
- The more likely the audience is to use the information to break the law, the more risky the publication
- If the security defect relates to digital rights management tools or other technological "locks" that control access to copyrighted works the advisory is more risky.
- There are no "whistleblower" protections under the applicable laws for security researchers. If the publication violates the law, or is proof of illegal research activities, the fact that the information obtained and reported was important for public safety it is not a defense.

In "The Laws of Full Disclosure" [78] SecurityFocus interviewed lawyers from twelve EU countries on the legal aspects of vulnerability disclosure. Most countries report to not have specific laws to handle "vulnerability disclosures". Whether or not such a disclosure is regarded as a criminal act depends on the country and the peculiarities of the case, e.g. whether the information was gained through illegal circumvention, the disclosure causes harm, the disclosure breaks an Non Disclosure Agreement (NDA), or whether the vulnerability is considered a trade secret. Generally, reporting that conforms with commonly accepted best practices is less likely to draw legal fire. However, disclosures outside of the "responsible disclosure" model may be both responsible and legal. Conversely, responsible disclosure may not protect you from being sued [48]. Different local implementations of the *Council of Europe Convention of Cybercrime (Art. 6, Sec. 2)* among European countries let us conclude that the concept of "vulnerability disclosure" will remain a debated topic in the near future.

Chapter 4

Security Information Providers (SIP)

The "disclosure debate" has taught us that timely and unrestricted access to security information is the best way to minimize the impact of vulnerabilities. The availability of timely security information from trusted and independent sources therefore plays a key role in analyzing and understanding the processes in the security ecosystem. As a result, analyzing the vulnerability lifecycle for a large number of vulnerabilities, we need access to accurate and unbiased information. Many private and government organizations specialize in collecting and publishing security information. In this chapter we identify and analyze viable security information sources for our research. The information provided by these organizations overlap and complement each other; there is no single best source. Therefore, rather than relying on a single source, our goal is to choose a number of suitable sources to be used for our research in order to minimize the risk of bias. We do not attempt to take all possible information sources into consideration; rather than being exhaustive, we choose a set of sources based on

criteria such as independence, accessibility, and available history of information. Throughout this dissertation we name these organizations Security Information Providers (SIP). We assess the quality, completeness, complementarity, independence, and timeliness of the information provided by the chosen SIPs. To understand the foundation of what we consider a vulnerability and how we correlate vulnerability information between different sources we introduce and explain the working of the Common Vulnerabilities and Exposures (CVE). Without a common language to correlate vulnerability-related information, it is difficult to manage the output from different information sources, security tools, vulnerability databases, and incident response teams. This unsatisfying situation led to the launch of CVE in 1999, a database of identifiers for publicly known security vulnerabilities that enjoys industry wide acceptance. Using CVE identifiers not only opens the door to correlate vulnerability information from disparate sources, through the industry wide acceptance it is ensured that any vulnerability of significance will eventually get listed in the CVE database.

4.1 Common Vulnerabilities and Exposures (CVE)

Common Vulnerabilities and Exposures (CVE) [79] was launched in 1999 as the first public database of computer vulnerability identifiers when most information security tools used their own databases with their own names for security vulnerabilities. MITRE [80], a non-profit organization of the U.S government chartered to work in the public interest, started the CVE list in cooperation with 19 major security organizations that made up the first CVE editorial board. The goal of the CVE database is to enhance information sharing and improve security tools. CVE now is a *de facto* industry standard that has achieved wide

acceptance in the security industry, academia, and a number of government organizations since its launch [81]. CVE identifiers are nowadays used in numerous security products, tools, and services around the world and numerous major OS vendors and other organizations include CVEs in their alerts and advisories to ensure that the international community benefits by having the vulnerability identifiers as soon as a problem is announced. From the original 321 entries in 1999, the CVE list has grown to over 27,000 entries as of January 2008. CVE provides the security community with:

- a comprehensive list of publicly known vulnerabilities
- a description of the vulnerability
- an analysis of the authenticity of newly published vulnerabilities
- an unique identifier for each vulnerability (e.g., CVE-2007-3039)

CVE Content

Each CVE identifier includes the CVE identifier number (e.g., CVE-2007-3039), a brief description of the security vulnerability, and any pertinent references to external sources (i.e., vulnerability reports, advisories, mailing list postings). Each reference used in CVE identifies the source and includes a well-defined identifier to facilitate searching on a source's Web site. As of January 1st, 2008 the CVE database listed 29,797 CVE entries with 158,779 external references to 77 different sources. Of these 29,797 entries 1084 (3.64%) were marked as *reserved*, 278 (0.93%) as *disputed*, and 272 (0.91%) as *rejected*. Candidate Numbering Authorities (CNA), organizations that frequently submit vulnerability information, may reserve a block of CVE identifiers in advance to streamline the handling of vulnerability

information [82]. In Table 4.1 we list the top ten most referenced information sources in the CVE database[1].

Key to the success and wide acceptance of CVE is the community approach. A number of organizations in the security community provide MITRE with vulnerability information that helps create new CVE identifiers. This information is provided to CVE in the form of *vulnerability submissions*. With multiple submissions from different organizations, CVE has a richer set of information to use when creating vulnerability identifiers. Since CVE does not rely on one single source, it has a better chance of identifying all publicly known security problems which then provides a more comprehensive set of vulnerability information for everyone.

CVE Editorial Board/Process

CVE has a content team whose primary task is to analyze, research, and process incoming vulnerability submissions. The team is led by the CVE editor, who is ultimately responsible for all CVE content. Submissions that pass the editorial board members review are accepted and entered into the CVE list (getting a CVE identifier assigned). If a submission is rejected, the editor announces the reason for rejection. The CVE editorial board includes numerous security organizations including vendors of commercial security tools, members of academia, research institutions, government agencies, and other prominent security experts. The board identifies vulnerabilities to be included through open and collaborative discussions and has made a commitment to make its discussions available to the public. MITRE moderates board discussions and provides guidance throughout the process to ensure that CVE serves the public interest. The board currently includes 31 experts from 22 different organizations [83].

[1] "SecurityFocus" stands for the SecurityFocus Vulnerability Database

Source	Referenced	Cumulative
Secunia	15.36%	15.36%
SecurityFocus	13.08%	28.44%
IBM ISS X-Force	12.36%	40.80%
BugTraq Mailing List	11.23%	52.03%
Miscellaneous	6.50%	58.53%
FrSIRT	6.47%	65.00%
OSVDB	5.29%	70.29%
SecurityTracker	4.05%	74.34%
Sreason	2.46%	76.80%
CERT	2.28%	79.08%

Table 4.1: *Top 10 most referenced sources in the CVE database as of January 1st, 2008. Details on sources in Appendix A.1.*

The content of the CVE database is available directly from the CVE Web site or through the MITRE partnership with the the National Vulnerability Database (NVD) [84], which is based on and synchronized with the CVE database.

National Vulnerability Database (NVD)

The National Vulnerability Database (NVD) is a database of cyber security vulnerabilities in information technology products that was developed by the National Institute of Standards and Technology (NIST). The NVD expands on the Internet Catalog (ICAT), a previous NIST project, that archived the vulnerabilities defined by the CVE database. Integrating all publicly available U.S. government vulnerability resources and including references to industry resources, the NVD is updated hourly to provide the latest information about vulnerabilities in

IT products. In addition to the content from CVE, NVD includes the following information:

- severity and impact of the vulnerability, including a CVSS[2] rating
- vendor name
- software name and version number
- vulnerability type

For this research we use the content of the NVD to determine the vendor and software affected by a vulnerability and the severity of the vulnerability. NVD provides severity rankings of "low", "medium", and "high" based on the CVSS [85] score of the vulnerability[3].

4.2 Information Sources

To compare the dynamics of thousands of vulnerabilities we examine the vulnerability lifecycle, normalized with respect to the disclosure time as shown in Fig. 3.1. Normalization with respect to the time of disclosure is rendered obvious as this is the first point in time the vulnerability becomes known to the public. Besides accurately determining the time of disclosure, we need a clear definition of what we consider a security vulnerability. Finally we need to identify viable information sources to gather the data needed for this research.

4.2.1 What is a Vulnerability?

As stated in Chapter 3, defining a vulnerability is a delicate business. In this research we are interested in the "real world"

[2]CVSS Common Vulnerability Scoring System
[3]low: 0.0-3.9, medium: 4.0-6.9, high: 7.0-10.0

impact of a security issue and not in the exact conformance of a technical specification. To accurately reflect the processes in the security ecosystem we delegate the decision on what counts as a vulnerability to the CVE consortium. According to CVE, a vulnerability is a mistake in software that can be directly used by an attacker to gain access to a system or network [24]. For this research we only consider security issues as a vulnerability if such an issue gets CVE identifier assigned. Delegation to define a vulnerability using CVE is a step towards precise quantification.

Definition 1 *For this research, only a security issue with an assigned CVE identifier is considered a* ***vulnerability****.*

On purpose this definition does not try to define certain technical properties of security issues as we primarily want to capture the real world impact of the security issues to accurately reflect the actual processes in the security ecosystem. Our definition therefore delegates the decision on what counts as a vulnerability to the CVE editorial board. Given the high acceptance of the CVE process in academia and the industry we assume that any security issue *of relevance* will eventually get a CVE number assigned as described in Section 4.1.

4.2.2 What is the Time of Disclosure?

In Chapter 2, we described that the *time of disclosure* of a vulnerability is defined differently among authors. It is most commonly referred to as a kind of *public disclosure* of *security information* by a *certain party*. To ensure the quality and availability of relevant security information, we propose a more strict definition of the disclosure time:

Definition 2 *The* ***time of disclosure*** $t_{discl}(v)$ *of a vulnerability* v *is the first time a vulnerability is described on a channel where*

the information disclosed *and the* information channel *publishing the vulnerability satisfy the following requirements:*

1. Free Access: *The disclosed vulnerability information is available to the public for free.*
2. Independence: *The vulnerability information is published by a widely accepted and independent source.*
3. Validation: *The vulnerability has undergone analysis by security experts such that risk rating information is included.*

Elaboration of the Requirements

1. *Free Access:*
   From a security perspective only a free and public disclosure of vulnerability information can ensure that all interested or concerned parties get the relevant security information. This requirement excludes expert vulnerability knowledge within closed user groups, or paid services that provide "advance notification" of vulnerabilities to their customers' only.
2. *Independence:*
   Only an information source independent of a vendor or government is unbiased and enables a fair dissemination of security-critical information[4]. This requirement implies the use of several sources to determine the time of disclosure as many of the organizations that publish security information are tied to a vendor or government. Using several sources minimizes the risk of bias. The disclosure time is the *first time* when *any of the sources* $s \in S$ reports the vulnerability:

$$t_{discl}(v) = min\{t_{discl}(v,s)\} \qquad s \in S \tag{4.1}$$

[4]In the remainder of this dissertation we will use the term *vendor* to name the manufacturer of the software for *commercial products*, *freeware*, and *open-source software* alike

We use the notation $t_{discl}(v,s)$ when the source s is relevant. For various reasons (including bias towards a vendors own products) different sources might publish a security advisory of vulnerability at different times (if at all). A channel is considered widely accepted only when it is an accepted source of security information in the security community (e.g., by having a delivered security information reliably over a long period of time). This requirement ensures the quality of the vulnerability information twofold: *independence* is a prerequisite to get unbiased and complete information while the *widely accepted source* builds confidence in the information delivered. We call viable sources of vulnerability information security information providers (SIP).

3. *Validation:*
   An analysis and risk rating of the vulnerability helps to ensure the quality and usability of the information disclosed. The mere discussion on a potential flaw in a mailing list or vague information from a vendor does therefore not qualify. The analysis must include enough details to enable a concerned user of the software to assess his individual risk or take immediate action to protect his assets.

The "disclosure debate" has taught us that timely and unrestricted access to security information is the best way to minimize the impact of vulnerabilities on social welfare. In combination, these three requirements ensure that the disclosure date reflects the first time when trusted, commonly understandable information about a new vulnerability is publicly available to everyone concerned. In the following we identify viable sources $s \in S$ to provide accurate timing information for the vulnerability disclosure.

4.2.3 Security Information Provider (SIP)

Rather than relying on a single source our goal is to choose a number of suitable sources to be used for our research in order to minimize the risk of bias. We do not attempt to take all possible information sources into consideration, rather than being exhaustive we choose a set of sources based on criteria such as independence, accessibility, and available history of information. To gather accurate timing information of the vulnerability lifecycle, an analysis based solely on the content of the CVE and NVD proved to be insufficient as these two initiatives rely largely on information provided by external sources. Therefore, and to satisfy Def. 2, we also examine the sources that feed CVE and NVD. To provide the data for the time of disclosure $t_{discl}(v)$ of vulnerabilities we analyzed security advisories of the seven "candidate sources" listed in Table 4.2. The goal of our analysis is to gather accurate information for the time of disclosure, it is not to rank these sources in any way.

Source	Abbrev	Country	Type	Start
US-CERT	CERT	US	gov	1988
SecurityFocus	SF	US	list	1996
IBM ISS X-Force	IBM-XF	US	list	1996
Secunia	Secunia	DK	pvt	2002
FrSIRT	FrSIRT	FR	pvt	2005
SecurityTracker	SecTrack	US	pvt	2001
SecurityWatch	SecWatch	US	pvt	2004

Table 4.2: *Candidate security information sources and type of organization: government sponsored (gov), publicly listed enterprise (list), private owned business (pvt)*

These sources were selected for practical reasons, mostly based on the prevalence of citations in CVE documents, the available history of security information, and the accessibility of their security advisories. In Table 4.1 we list the top ten most referenced sources in the CVE list, which together account for 79% of all references in CVE. SecurityFocus, owned by Symantec, stands for the "SecurityFocus Vulnerability Database" which is not equal to the BugTraq mailing list owned by SecurityFocus nor the alerts of Symantec's security response team. SecurityWatch was included as it was frequently referenced in the security advisories of the other sources. Note that FrSIRT is a privately held company independent of the French government, despite its name *French Security Incident Response Team.* A short introduction to each source is given in Appendix A.1. In the following of this chapter we examine the suitability of these services to act as a source of the disclosure date for our research. In Section 4.5 we discuss the role of these sources in the security ecosystem. We commonly refer to these sources using the abbreviations as of Table 4.2. We compare the completeness, working patterns and timeliness of the information published by these services since 2004. To shed a light on the dissemination of insecurity information (exploit material), we also include three well known exploit archives in our study for comparison.

4.2.4 Exploit Archives

To get a general view we are not only interested how *security* information is disseminated, we are also interested how *insecurity* information is spread. While cyber-criminals are not known to share information publicly, we can estimate the availability of insecurity information through the analysis of public exploit archives. The public availability of exploit material is a minimum estimator for the security information available to cyber-criminals. Some security sources include links to exploit material

in the ”external references” section of their advisories. The three most referenced exploit archives are *Milw0rm* [86], *Packetstorm* [87], and *SecurityVulns* [88]. We use these sources to compare the daily and weekly working patterns of exploit publications with security advisory publications.

4.3 Methodology

To identify viable sources to act as SIPs we analyze and compare the advisory publication performance between the selected candidate sources since 2004. Our methodology consists of three major phases:

1. Enumeration and collection of all security advisories published by the candidate sources since 2004.
2. High resolution monitoring of advisory publications since August 2006.
3. Correlation of the information gained in Phase 1 and Phase 2.

4.3.1 Data Collection (Phase 1)

To get an impression of the extent of security advisory publications, we first download all security advisories published by a given source since 2004. A complete list of advisory URLs to be downloaded is compiled from several sources:

- References in the NVD and the CVE database.
- References in the *Archive* section of the source where they host lists to all their past advisories.
- Cross references in the advisories of other sources.

- Enumeration of advisory URLs in case the format followed a predictable pattern (*e.g., sequential advisory IDs*).

After the download of these advisories representing more than 200,000 documents, a parser extracts the following information (if available) from these documents:

- The identification of the advisory (*e.g., Bugtraq-ID, X-Force-ID, Secunia-ID*).
- The disclosure date (publication date) of the advisory.
- The risk rating of the vulnerability.
- The CVE, or list of CVEs in the advisory[5].
- URLs to external references of other security or exploit sites.

The output of the parser is fed into a database for analysis and correlation.

4.3.2 High Resolution Monitoring (Phase 2)

Downloading and parsing security advisories from various sources, as done in Phase 1, supplies the publication date with a resolution of one calendar day only. This is the publication date as stated by the source, which might could not correspond to the real date the advisory was made available to the public. To get the publication time independently and on a higher resolution we deployed a monitoring Web spider. This Web spider monitors and analyzes the *entry page* of the specified Web sites twice an hour in order to identify and timestamp the appearance of new security advisories. To identify new advisories, a parser

[5] A security advisory can include more than one CVE, e.g., if the advisory describes more than one vulnerability

extracts all URLs found in the entry page and compares it to the list of URLs from the last download. Newly found URLs that match the format of URLs to security advisories (or exploits) are timestamped and logged for later analysis and correlation. We started monitoring the Web sites of our candidate sources and exploit archives (as listed in Section 4.2.3) in August 2006. The output of this phase is also fed into the database.

4.3.3 Correlation (Phase 3)

The correlation of the information from Phase 1 and Phase 2 is a two step approach The first step is proper identification of vulnerabilities across different sources; The second step is correlation of vulnerabilities with the data from the monitoring spider.

For reasons stated in the beginning of the chapter we only consider vulnerabilities having a CVE identifier assigned. In most cases, a CVE identifier is found in the advisory itself (where the advisory refers to the corresponding CVE identifier). However, in many instances the reference to the CVE identifier was entered well after the initial publication of an advisory by a candidate source. This is the case when a candidate source reports a new vulnerability that has not yet be given an CVE identifier, e.g. when reporting a vulnerability based on the analysis of a zero-day exploit found in the wild. The candidate source publishes the security advisory and reports it to the CVE editorial board at the same time. Later, when a CVE identifier is assigned to the vulnerability, the original security advisory is updated and the respective CVE identifier added. Therefore, Phase 1 and Phase 2 are executed concurrently. In an initial run of Phase 1 we downloaded all advisories published before the start of the monitoring spider. The monitoring spider described in Phase 2 does not download the advisories but collects the URLs to new advisories, together with a timestamp. At a later

time we download the content of these advisories to capture the cases when a CVE identifier was added later. Nevertheless, some candidate sources were found not to add CVE identifiers in their advisories on a regular basis. To capture these cases, we used cross references in NVD and CVE documents for the correlation of CVE identifier to advisory (e.g., over 40% of all the CVEs of SecurityFocus were assigned this way). Normalization of all URLs used for cross referencing was necessary as in some instances the format of these references changed substantially over time.

The output of this step is a set of CVE identifiers with several advisories assigned from different sources. Let S be the set of candidate sources as of Section 4.2.4, E be the set of exploit sources as of Section 4.2.4, and V be the set of vulnerabilities captured in Phase 1 and 2. We then denote the disclosure time of vulnerability $v \in V$ reported by source $s \in S$ as

$$t_{discl}(v,s) \rightarrow time \qquad v \in V,\ s \in S \tag{4.2}$$

If several advisories of the *same* source refer to the same vulnerability v, then $t_{discl}(v,s)$ represents the publication time of the first advisory. Correspondingly, the time of exploit availability of vulnerability $v \in V$ reported by exploit source $e \in E$ is

$$t_{explo}(v,e) \rightarrow time \qquad v \in V,\ e \in E \tag{4.3}$$

The correlation using CVE identifiers allows the comparison of security advisories from multiple sources relating to the same vulnerability.

4.4 Analysis of Security Information Sources

Competition in the security information business is fierce, and there is no shortage of information suppliers. We analyze and

compare the performance of the candidate sources identified in Section 4.2.3. In the remainder of the chapter we examine the following questions:

- *Where is the most appropriate information about vulnerabilities found?*
- *How do these sources support the processes of the security ecosystem presented in Section 3.2?*

To answer these questions, we analyze different aspects of the problem:

- What source (or what combination of sources) provides the best coverage of vulnerabilities reported to the total number of vulnerabilities?
- How well are CVEs covered by different sources?
- Which are the fastest sources to report new vulnerabilities?
- What are the working patterns (i.e., are there certain schedules of deliveries)?

4.4.1 Completeness of Disclosures

To get insight in the extent of security information coverage, we first count the number of advisories published by each source from January 1^{st}, 2004 to December 31^{st}, 2007. In Table 4.3 we list the total number of *advisories published* by a given source, including advisories without a CVE assigned. In Table 4.4 we list the total number of *unique CVEs* covered by these advisories for each candidate source. For each source the year of publication is taken from the publication date entry of the respective security advisory. For every year we include the vulnerability counts of the NVD (based on the NVD publication date) for reference.

Note that *FrSIRT* started operation only in 2005. The number of security advisories published per source does not necessarily equal the number of CVEs reported. This becomes evident when comparing the content of Table 4.3, Table 4.4, and Fig. 4.1, where we visualize the total number of advisories and the share of unique CVEs for 2006 and 2007.

A CVE identifier can be covered by more than one security advisory of the same source while other advisories cannot be related to any CVE identifier at all. It all depends on the policy of what a specific source counts as noteworthy to release as an advisory. For example, at the patch day of a big vendor an overview advisory is published, containing all CVEs of the vulnerabilities patched. Very likely, many of these vulnerabilities and CVEs have already been reported in several individual advisories prior to the patch day, and some vulnerabilities will be reported individually after the event. *Therefore, to prevent redundancies and discrepancies in vulnerability counting, we rely on Definition 1 and only consider vulnerabilities that have a CVE identifier assigned.* Simple vulnerability counts based on the number of security advisories are meaningless.

Fig. 4.1 shows that for a complete coverage it is not advisable to rely solely on one single source. Of the seven sources examined, only IBM X-Force covered more than 80% of CVEs in 2007, and three sources even scored below 50%. We extend the counts from Table 4.4 and include combinations of any two of (*IBM X-Force, SecurityFocus, Secunia, FrSirt*) for the year 2007. For this comparison we excluded the three sources that scored below 50% as of Fig. 4.1, they do not add substantially to the other sources due to correlation. A total of 6,532 (100%) unique

Source	2004	2005	2006	2007
CERT	350	315	505	340
IBM-XF	2,810	4,719	7,060	6,312
SF	2,368	3,500	5,564	4,941
Secunia	4,150	8,523	10,794	9,231
FrSirt	-	3,072	8,311	6,337
SecTrack	1,555	1,961	2,389	1,793
SecWatch	536	1,523	1,343	1,291
NVD	2,450	4,928	6,600	6,532

Table 4.3: *Number of all security advisories published (including those* without *a CVE) per source and year.*

Source	2004	2005	2006	2007
CERT	321	299	480	330
IBM-XF	2,600	4,401	6,672	6,022
SF	2,303	3,302	5,386	4,797
Secunia	2,063	4,022	5,754	4,535
FrSIRT	-	2,282	5,019	3,842
SecTrack	1,488	1,840	2,162	1,665
SecWatch	429	1,216	1,126	1,098
NVD	2,450	4,928	6,600	6,532

Table 4.4: *Number of* unique CVEs *covered by advisories of given source and year.*

CVEs were released in 2007 according to the NVD (based on the NVD publication date). Table 4.5 shows that the best coverage one can get from a single source is 92% of the CVEs released that year. However, when the information of two providers are combined, we exceed 95% and even get up to 99% completeness. We find that for a complete coverage of security information, it is advisable to consult at least two different sources. Not only

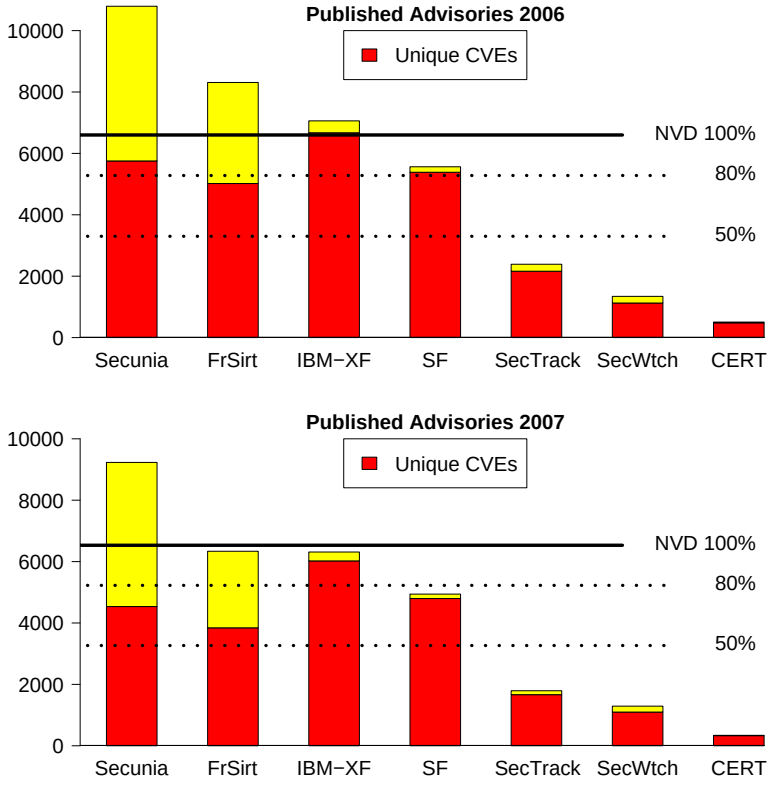

Figure 4.1: *Total advisories and share of unique CVEs for 2006 and 2007*

do the sources complement each other, relying on several sources provides more robust information and neutralizes a potential bias of a single source. By way of example, in 2007 the best coverage achieved within the seven sources examined would be by the combination of IBM X-Force and Secunia. This analysis is aimed at gathering information for the maximum number vulnerabilities for our research of the security ecosystem. Publishing more is not

necessarily better, it depends on the policy of what a specific SIP publishes. E.g. a source might only publish information of high risk vulnerabilities of specific vendors which is of value for their customer base.

Source	IBM-XF	SF	Secunia	FrSIRT
IBM-XF	6,022 92%	6,264 95%	6,437 99%	6,416 98%
SF		4,797 73%	5,802 89%	5,637 86%
Secunia			4,535 69%	5,042 77%
FrSirt				3,842 59%

Table 4.5: *Number and percentage of unique CVEs covered by any combination of two sources.*

4.4.2 Disclosure Working Patterns

Time is of the essence in security. We therefore examine the publication patterns by day of week and by the hour during the day for the subset of vulnerabilities $v \in V' \subset V$ published in 2007 for which we have high resolution timing information from our monitoring agent. We limited the scope of this analysis to a full calendar year to neutralize potential seasonality effects in vulnerability publications. We compare the timeliness of publication of the sources $s \in S$ and include the exploit archives $e \in E$ for comparison.

Organization	Location	Timezone
CERT	USA, Pittsburgh PA	UTC-5
IBM-XF	USA, Atlanta GA	UTC-5
Securityfocus	USA, Cupertino CA	UTC-8
Secunia	Denmark, Copenhagen	UTC+1
FrSirt	France, Montpellier	UTC+1
SecTrack	USA, Silver Spring MD	UTC-5
SecWatch	UK, Exeter	UTC+0
Milw0rm	USA, Waco TX	UTC-6
SecurityVulns	Russia, Nizhny Novgorod	UTC+3
PacketStorm	N/A	N/A

Table 4.6: *Location and time zone of information sources.*

Working Hours

In Fig. 4.2 we plot the distribution of the disclosure time during the 24 hours of the day. All time information is normalized to UTC. Analyzing the intra-day distributions shown in Fig. 4.2 we found it nicely correlates with the geographical location of the organizations, listed in Table 4.6. *IBM X-Force*, *SecurityFocus*, *SecurityWatch*, and *CERT* operate in US timezones. *Secunia* and *FrSIRT* operate in Europe while *PacketStorm* appears to operate or receive its exploit contributions from a Far East timezone. This suggests that we observe the working patterns during the day according to the local time zone of the organization. For all sources except *SecurityTracker* and *SecurityVulns*, we find a clear pattern of working and non working hours.

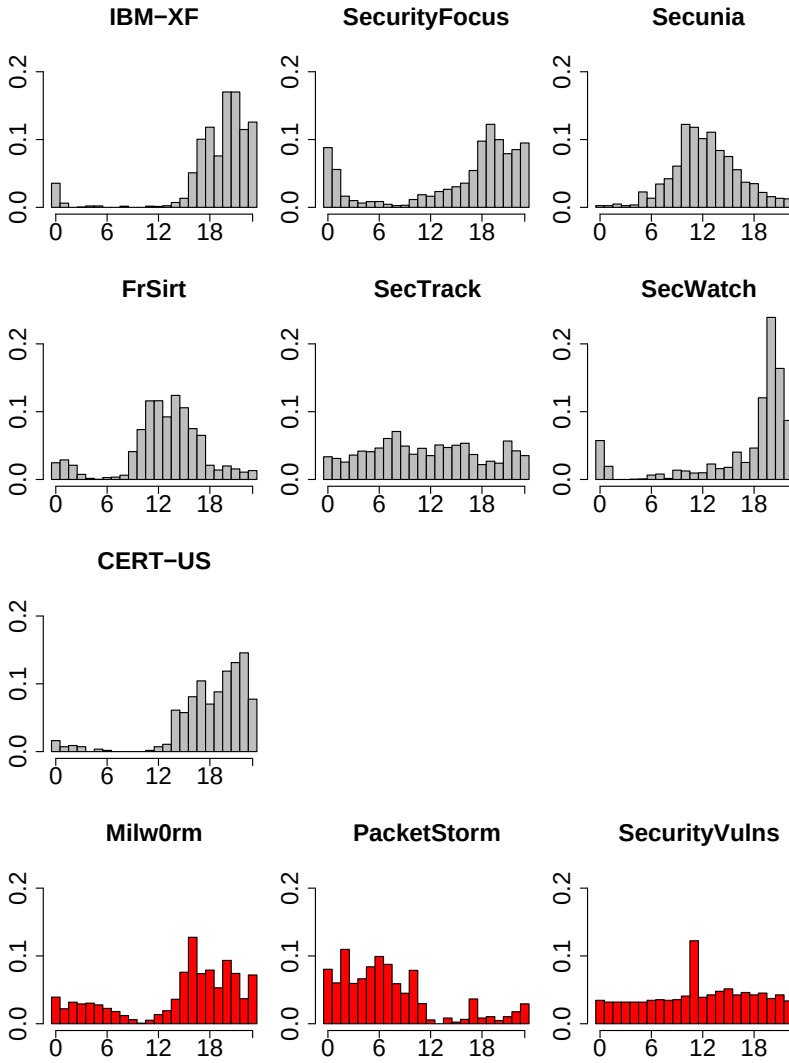

Figure 4.2: *Distribution of advisory and exploit disclosures by hour of the day, timezone UTC.*

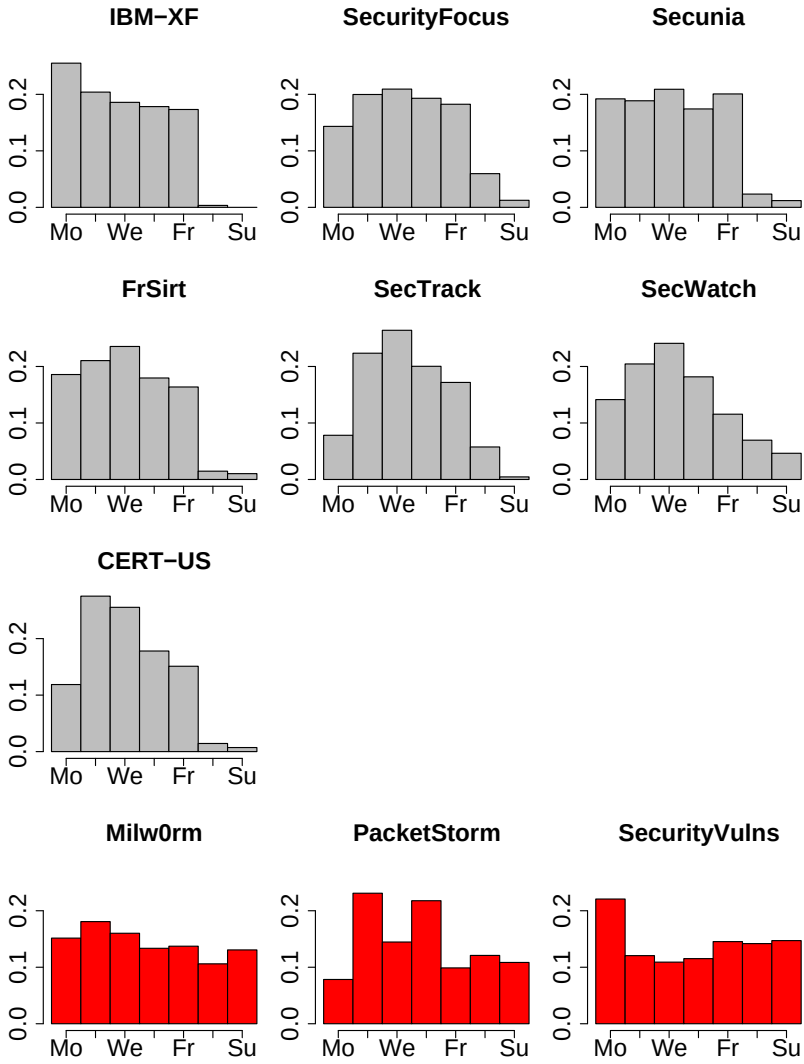

Figure 4.3: *Distribution of advisory and exploit disclosures by day of week.*

We observe a rather uniform distribution in the hourly distribution of *SecTrack* and *SecurityVulns*. Such strong correlation is often caused by the deployment of automated information retrieval tools. As of its service description, *SecTrack* does not perform its own research and relies heavily on automated information retrieval tools (See Appendix A.1). The peak at 11h UTC in the hourly distribution of *SecurityVulns* could be the result of a daily batch update of the sites content. The observation of intra-day working patterns and correlation with time zones is an indication that the sources S (except SecurityTracker) actually process security information and not merely copy from another source using automated tools.

Working Days

In Fig. 4.3 we examine the distribution of advisory and exploit publications by the day of the week. We find that the seven security information providers $s \in S$ follow a clear workday/weekend pattern of disclosures, with few or no security advisories being disclosed during the the weekend. *SecurityTracker*, which we believe to operate mainly by using automated information retrieval tools, also drops at the weekend. This is the result of the drop of its information sources. The weekly pattern of the security information providers contrasts to the pattern of exploit material publications. *Milw0rm*, *Packetstorm*, and *SecurityVulns* show an almost uniform disclosure rate throughout the week as shown in Fig. 4.3. When new exploits are released over the weekend, there will be likely a longer delay until the public has access to this information through the free services offered by SIPs. Note that most customer organizations typically don't have security teams working weekends - so they wouldn't do act on the information even if it was available over the weekend. This is actually one of the reasons why organizations use managed

security services (MSS) offerings over weekends and other "out of hours" monitoring services.

4.4.3 Disclosure Performance Comparison

In this section, we compare the publication performance of the candidate sources within 48 hours of the first report of a vulnerability. We again analyze the subset $v \in V' \subset V$ of vulnerabilities published in 2007 for which we have high resolution timing data from Phase 2. For each vulnerability v, we first determine the disclosure time $t_{discl}(v)$ when v was first reported by *any* of the sources $s \in S$ using Eq. 4.1:

$$t_{discl}(v) = min\{t_{discl}(v,s)\} \qquad v \in V',\ s \in S$$

Then we examine the distribution of the publication delay Δt_s of source s with respect to the disclosure time $t_{discl}(v)$:

$$\Delta t_s(v) = t_{discl}(v,s) - t_{discl}(v) \tag{4.4}$$

We examine the timeliness of a given source examining the cumulative probability $\mathcal{P}_{\leq}(\Delta t_s \leq x)$. $\mathcal{P}_{\leq}(\Delta t_s \leq x)$ gives the fraction of vulnerabilities source s publishes withing x hours after the first publication[6]. We use the empirical cumulative distribution function (ECDF) to determine $\mathcal{P}_{\leq}(\Delta t_s \leq x)$. The $ecdf_s(x)$ of $\Delta t_s(v)$ for n vulnerabilities of source s is

$$\begin{aligned} \mathcal{P}_{\leq}(\Delta t_s \leq x) &= ecdf_s(x) \\ &= \left|\left\{ v \in V \mid \Delta t_s(v) \leq x \right\}\right| \end{aligned} \tag{4.5}$$

In Fig. 4.4 we plot the cumulative distribution of the short time dynamics of vulnerabilities disclosed by a given source s within time $x = 0 - 48h$ after the first disclosure. Note that

[6] $\mathcal{P}_{\leq}(X \leq x)$ denotes the cumulative distribution function of random variable X [89]

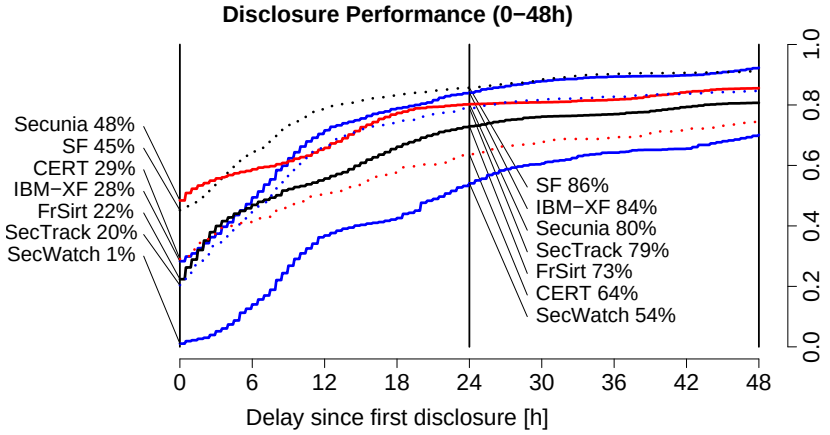

Figure 4.4: *Share of publications within 0 to 48h after the first report of a vulnerability.*

several sources can report the same vulnerability concurrently. This is the case when more than one source publishes the same vulnerability within our 30 minutes sampling resolution. Most often, *Secunia* and *SecurityFocus* are the first to disclose a new vulnerability with 48% of the Secunia advisories and 45% of the SecurityFocus advisories achieving $\Delta t_s = 0$.

All sources have a share of at least 20% "first to report vulnerabilities", except *SecWatch* with only 1%. Within 24 hours every source published more than 50% of their advisories and all but two published even more than 75% of their advisories in this time. This is a remarkably high publication performance within 24 hours of the first publication. At 24 hours, *SecurityFocus* and *IBM-XF* lead with 86% and 84% share, closely followed by *SecTrack* and *Secunia* with 80% and 79%. The effect of different timezones is visible in the low frequency modulation of the different curves' slopes in Fig. 4.4 within the first 24 hours.

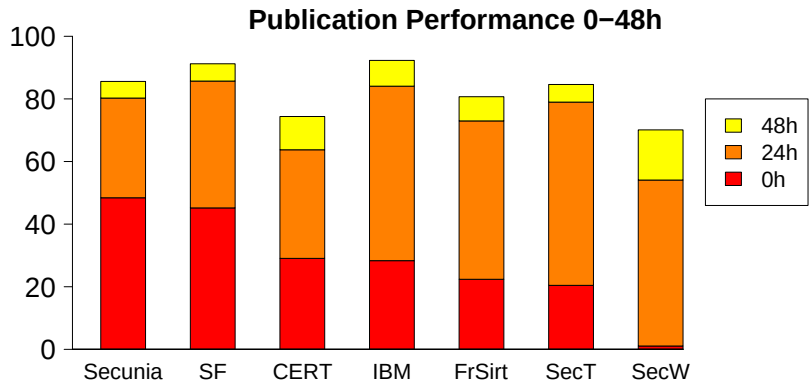

Figure 4.5: *Share of publications within 0h, 24h, and 48h after the first report of a vulnerability.*

In Fig. 4.5 we separately plot the values of $\mathcal{P}_{\leq}(\Delta t_s \leq x)$ for $x \in \{0h, 24h, 48h\}$ to visualize the results at these times per source. The high dynamics in the first 24 hours is clearly visible. *SecWatch* and *CERT*, both covering a comparably low number of vulnerabilities compared to the other sources (see Table 4.4), fall consistently behind with 63% and 54% share at 24 hours. We see that analyzing fewer vulnerabilities does not mean to be faster. With the exception of *SecurityFocus* and *SecurityTracker*, all security information providers include a risk rating with their security advisory. However, the risk classifications in use differ among the sources, as listed in Appendix A.1. To analyze the impact of the risk level of a vulnerability on the timeliness of publication we use the three level risk rating of NVD. NVD rates every CVE as either a *high*, *medium*, or *low* risk vulnerability. We found that the timeliness of disclosure does not depend on the assigned risk level of the vulnerability. There is no significant deviation in publication speed between the sources based on the risk level of a vulnerability.

4.5 The Role of Security Information Providers

We analyzed the security information publications of seven different organizations, one of which is a government sponsored agency (CERT) while the others are privately and publicly held companies of various sizes operating with different business models in three different countries on two continents. Two companies are publicly held and among the biggest producers of security software worldwide accompanied with considerable research resources (IBM, Symantec), two privately held companies are fully dedicated to the dissemination of security information also running their own security research (Secunia, FrSirt), one solely collects security information gathered from others without research (SecurityTracker) and one went out of business as of May 2008, to reopen in November 2008 as a mere search engine of other security advisories (SecWatch).

4.5.1 Competitive Market

It is clear that it is next to impossible to formally prove the independence of multiple security information sources. However, the variety of players and their business models together with the analysis of the *Completeness of Disclosures*, the *Disclosure Working Patterns*, and the *Disclosure Performance Comparison* in the previous section suggest that we observed at least several independent players acting in a competitive marketplace. Generally, competition in a open market enables and strengthens independence. All but two have their own security research staff, work according to typical working patterns, and differ in the set of vulnerabilities they disclosed. To get coverage of all vulnerabilities disclosed in a year we have to look at more than one single source. With one exception (SecWatch), we found that

all sources are first contributors and there is no evidence that everyone simply copies their information from one single source.

We conclude that we observe a healthy and highly competitive market between the different of security information providers. This market ensures that the public has access to timely and accurate security information. No single source covered all vulnerabilities disclosed in a given year, in order to get a complete coverage one needs to track the advisories of more than one source.

4.5.2 The Role of Independent Security Information Providers

The "disclosure debate" taught us that timely and unrestricted access to security information is the best way to minimize the impact of vulnerabilities on social welfare. Recent research in security economics reveals that security is not only a technology problem. The incentives of the security environment, as perceived by the players, have a major impact on the resulting processes and the composition of the security ecosystem. Therefore, the public availability of commonly understandable security information from independent and trusted sources, SIPs, is a key component in the security ecosystem. The primary and most visible impact of SIPs is providing the public with timely and accurate security information. The indirect and less visible impact is the regime of incentives that are actually built and established by the presence and operation of SIPs in the security ecosystem as depicted in Fig. 3.2:

- Having several independent SIPs is the best guarantee that security issues will eventually surface. SIPs actively research the security and insecurity scene and publicize their finding of new vulnerabilities in a responsible way. Every vulnerability discovered this way is one less vulnerability to

be first fully exploited by cyber-criminals, or kept hidden by the vendor.

- The threat of publishing a vulnerability is almost as good as actually publishing it. The presence of trusted and widely heard SIPs support the process of *responsible disclosure*. When a new vulnerability is brought to the attention of a vendor, the vendor is fully aware that he has no room to downplay or ignore the severity of the vulnerability, whoever the discoverer. We further analyze this effect in Chapter 6 in a case study of Microsoft and Apple.

We conclude that the diversity and choice of available SIPs operating in a competitive marketplace is clearly preferred over a single government sponsored agency providing security information [90]. Market forces and multiple SIPs operating in different countries are the best guarantee for independent, accurate and timely security information. SIPs effectively and efficiently monitor the (in)security scene, new security issues are quickly released as security advisories to the public.

4.5.3 Conclusion

Independent and trusted SIPs act like the free press in an open society: they are efficient watchdogs to expose important issues to the public.

- Issues addressed are hard to be denied by either the vendor or a government.
- The presence of several independent security information providers is key to establish healthy incentives in the security ecosystem.
- This is an essential role for the well-being and functioning of the security ecosystem.

Chapter 5

Dynamics of (In)Security

5.1 Introduction

The area of quantitative studies of the security ecosystem at large is still in its infancy. The results of the classical measurement of the cumulative number of disclosed vulnerabilities over time is an interesting factor of the increasing risk exposure, but it is of limited use in understanding the underlying processes. In this chapter, we analyze the state and the evolution of the security ecosystem over the past decade based on an empirical dataset of more than 27,000 vulnerabilities disclosed between 1996 and 2008. We look at the evolution of the dynamics between security *(availability of patches)* and insecurity *(availability of exploits)*, normalized to the time of vulnerability disclosure. The availability of an exploit poses a security threat whereas the availability of a patch neutralizes this threat if the patch gets installed on the vulnerable system. Assuming that both the exploit and the patch work as intended by the respective originator, the resulting security risk for software users will depend strongly on the timing or dynamics of the availability of those. We use aggregated information of the vulnerability lifecycle events to measure the

prevalence of the different processes in the security ecosystem introduced in Chapter 3 and illustrated in Fig. 3.2 on Page 34, which also depicts the intimate relation between the vulnerability lifecycle events and the processes in the security ecosystem. We measure the current state and identify global trends.

The data for this research is gathered exclusively from publicly available sources. To create a comprehensive vulnerability database we download, parse, and correlate the information of well over 200,000 individual security bulletins of various sources, applying the methodology introduced in Phase 1 and Phase 3 of Section 4.3 in the previous chapter. Based on this vulnerability database we extract the needed information of the lifecycle events for our analysis. In Chapter 6 we further analyze the patch release dynamics of specific vendors. The time of patch installation cannot be determined using our vulnerability database. To complete the picture we present a study of the patch installation dynamics of Web browsers based on a different dataset in Chapter 7.

5.2 Static Analysis

Before we examine the dynamics of the vulnerability lifecycle, we look at the total number of vulnerabilities disclosed over the last decade and how these vulnerabilities are distributed among different vendors and risk classes. In Fig. 5.1 we plot the cumulative number of vulnerabilities disclosed since 1996 and the number of disclosures by year and risk rating, based on the content of our vulnerability database. The risk classification is taken from the National Vulnerability Database (NVD), introduced in Section 4.1. NVD also provides the mapping of a specific vulnerability to the vendor of the affected software and the name of the product. From 1996 to 2007 NVD lists 6,162 different vendors affected by vulnerabilities disclosed in

this period. Consistently, most vulnerabilities are classified as either "high" or "medium" risk and through 2006 we see a steady increase in the number of vulnerabilities disclosed per year. To get more insight we examine how these vulnerabilities are distributed among different software vendors. The distribution of these vulnerabilities among the affected vendors is depicted in Fig. 5.2, and Fig. 5.3.

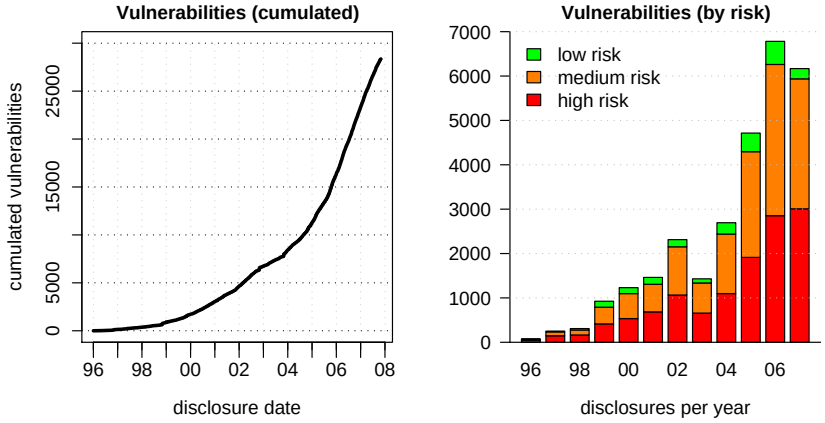

Figure 5.1: *Vulnerability disclosures 1996-2007.*

The left panel in Fig. 5.2 shows the vendor vs. vulnerabilities distribution per year in log-log scale. Only a few vendors account for most vulnerabilities published in a given year, so we observe a skewed, power-law like distribution. This fact is further illustrated in the right panel of Fig. 5.2 where we plot the combined share of the *top-N* vendors (affected by vulnerabilities) per year since 1998 for $N \in \{1, 5, 10, 20, 50, 100\}$. E.g. only $N = 10$ (or 0.04%) of the 2,491 vendors of vulnerable software in 2007 are responsible for 20% of the reported vulnerabilities in that year. Fig. 5.3 lists the names of the *top-ten* vendors from 2000 to 2007. From this analysis we observe that most of the vulnerabilities published in any given year affect well known

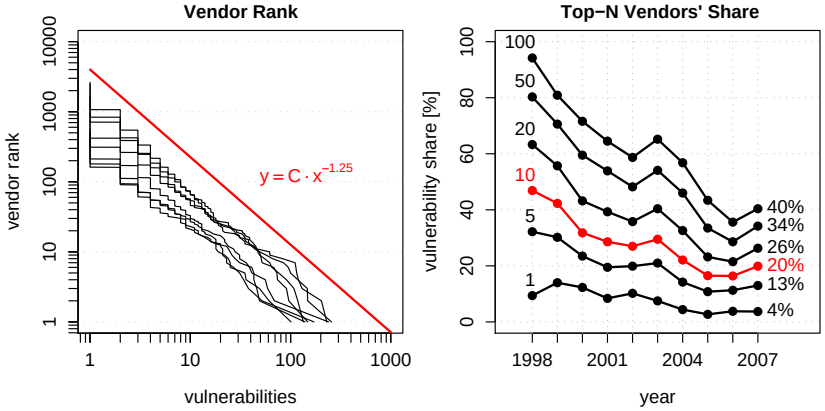

Figure 5.2: *Distribution of vulnerabilities among different vendors. Source NVD.*

commercial and open-source software vendors. These vendors produce the majority of software products in daily use at home within business. As a result most of the vulnerabilities disclosed are of relevance to the majority of users. The decreasing share of the *top-N* vendors can be attributed to the combined effect of (a) the improved QA[1] and testing processes deployed by the largest software vendors (which have removed many of the vulnerabilities that can be discovered by "newbie" researchers) and (b) the fact that there are more and more software packages released every year by new software companies – many of them offer easier pickings for security researchers and automated discovery tools [91]. However, these numbers contain no information about the duration for which we are exposed to the vulnerability, when the vulnerability has become known to the public, or when a patch was made available. When the vulnerability *discovery time*, *disclosure time*, and the *patch availability time* are known, we can determine the risk exposure time.

[1] Quality Assurance

2000	2001	2002	2003	2004	2005	2006	2007
Microsoft	Microsoft	Microsoft	Microsoft	Microsoft	Microsoft	Microsoft	Microsoft
Red Hat	Sun	Cisco	Sun	Gentoo	Apple	Apple	Apple
HP	Cisco	HP	Apple	Red Hat	Linux	Oracle	IBM
FreeBSD	IBM	Sun	IBM	Apple	Mozilla	Mozilla	Oracle
Sun	Red Hat	Oracle	Red Hat	Linux	Sun	Linux	PHP
IBM	HP	IBM	SGI	SuSE	IBM	IBM	Sun
Debian	Mandrake	SGI	HP	SGI	Oracle	Sun	Cisco
Mandrake	Oracle	Apache	Apache	Mozilla	Red Hat	Cisco	Mozilla
OpenBSD	FreeBSD	FreeBSD	Cisco	Mandrake	Ethereal	Joomla	HP
SuSE	Debian	Mozilla	Linux	Sun	SuSE	Novell	Linux

Figure 5.3: *List of the top-ten vendors affected by most vulnerabilities based upon percentage of total vulnerabilities in a year per year from 2000 to 2007. Source NVD.*

5.3 Dynamics Analysis

The basis for our measurements is the vulnerability lifecycle introduced in Chapter 3 and visualized in Fig. 3.1 on Page 24. Generally, we are interested in the distribution of the timing and the sequence of the events *discovery*, *exploit availability*, and *patch availability* of the vulnerability lifecycle at large. The reference point of each vulnerability is the *time of disclosure* which is known for all vulnerabilities, as defined in Section 4.2. This information gives insight into the processes and the state of the security ecosystem. In the following sections we present empirical results, discuss, and interpret these lifecycle events individually.

5.3.1 Methodology

Throughout this chapter, V denotes the set of all vulnerabilities (CVEs) released before January 1st, 2008. For all vulnerabilities $v \in V$ we know $t_{discl}(v)$, the time of the vulnerability disclosure taken from the fastest SIPs reporting this CVE with a resolution

of one calendar day, as described in Section 4.2. As of Eq. 3.5 the time of disclosure is denoted:

$$t_{discl}(v) \rightarrow time \qquad v \in V$$

The analysis of this chapter is based on the content of our vulnerability database. In Fig. 5.4 we plot the fraction of vulnerabilities for which we found the time of discovery $|V_{disco}|/|V|$, time of exploit availability $|V_{explo}|/|V|$, and the time of patch availability $|V_{patch}|/|V|$ of every year from 2000 to 2007. The absolute number of vulnerabilities disclosed in a given year (100%) is visibile in Fig. 5.1. In the following sections we individually discuss the dynamics of vulnerability *discovery*, *exploit availability*, and *patch availability* and describe the data sources used to build V_{disco}, V_{explo}, and V_{patch}.

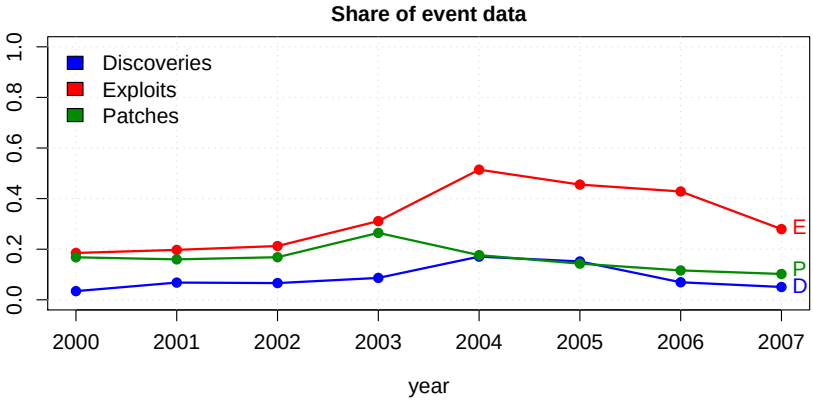

Figure 5.4: *Share of observed events within all vulnerabilities from 2000 to 2007*

We examine the vulnerability lifecycle by looking at how the time of the events $\alpha \in E = \{disco, explo, patch\}$ relate to the respective disclosure time $t_{discl}(v)$ of the vulnerability. For all vulnerabilities from 2000 to 2007 and each type of event, we

present a scatter plot, the associated distribution function, and yearly summaries to evaluate the evolution and identify trends. These plots are discussed in detail in the following sections. Normalization of the vulnerability lifecycle events with respect to the disclosure time is key to evaluate the aggregated dynamics of thousands of vulnerabilities. As of Eq. 3.7 and Eq. 3.8 we build Δt_{disco}, Δt_{explo}, and Δt_{patch} as follows

$$\Delta t_\alpha(v) = t_\alpha(v) - t_{discl}(v) \qquad \alpha \in E,\ v \in V_\alpha \tag{5.1}$$

Essentially $\Delta t_\alpha(v)$ returns the number of days event $\alpha \in E$ happened *before* or *after* the disclosure of vulnerability v:

$$sgn(\Delta t_\alpha(v)) = \begin{cases} -1 & \alpha \text{ occurs } \textit{before} \text{ disclosure} \\ 0 & \alpha \text{ occurs on } \textit{same day} \text{ as disclosure} \\ 1 & \alpha \text{ occurs } \textit{after} \text{ disclosure} \end{cases}$$

Δt_{disco} is an estimator of the "pre-disclosure" risk and Δt_{patch} is an estimator of the "post-disclosure" risk period as introduced in Section 3.1.3.

Scatter plots

We first use scatter plots of Δt_α to visualize the distribution and the evolution of events $\alpha \in E$ over the last eight years. In the scatter plots of Fig. 5.5, Fig. 5.7, and Fig. 5.9, each point $P_\alpha(v)$ of event α is built according to

$$P_\alpha(v) \rightarrow (x,y) \qquad \begin{cases} x = t_{discl}(v) \\ y = \Delta t_\alpha(v) \end{cases} \qquad \alpha \in E,\ v \in V_\alpha \tag{5.2}$$

In all scatter plots, the x-axis is the calendar day of the disclosure of vulnerability v. The y-axis represents the time difference to the disclosure of vulnerability v. Events with $y > 0$ occurred *after* the disclosure, events with $y < 0$ occurred *before* the disclosure of the vulnerability v plotted.

Distribution function

To further analyze the dynamics, we plot and discuss the cumulative distribution $\mathcal{P}_{\leq}(X \leq x)$ of the same data used to generate the scatter plots. Throughout this dissertation we use the notation $\mathcal{P}_{\leq}(X \leq x)$ to denote the cumulative distribution of random variable X as of Sornette [89]. We calculate $\mathcal{P}_{\leq}(X \leq x)$ by means of the *empirical cumulative distribution function (ECDF)*. The $ecdf_{\alpha}(x)$ of $\Delta t_{\alpha}(v)$ for $n = |V_{\alpha}|$ vulnerabilities $v \in V_{\alpha}$ of event $\alpha \in E$ is

$$\begin{aligned} \mathcal{P}_{\leq}(X \leq x) &= ecdf_{\alpha}(x) \\ &= \left| \{ v \in V_{\alpha} \mid \Delta t_{\alpha}(v) \leq x \} \right| \end{aligned} \tag{5.3}$$

In Fig. 5.6, Fig. 5.8, and Fig. 5.10 we plot the $ecdf_{\alpha}(x)$ for discovery, exploit, and patch availability for the range $x = \pm 400$ days. These plots give insight in to the aggregated dynamics of the vulnerability lifecycle.

5.4 Discovery Dynamics

Usually the time of discovery of a vulnerability is not publicly known until *after* its disclosure. Indeed, for many vulnerabilities the time of discovery will never be known or reported to the public, depending on the motives of the discoverer. Again, cybercriminals won't provide this information to the public. However, there are several sources from which we can derive the time of vulnerability discovery. One source is the Open Source Vulnerability Database (OSVDB) [92], another source are the security bulletins of commercial vulnerability markets. A short description of these sources is provided in Appendix A.1. We introduced commercial vulnerability markets as "white markets" in the discussion of the processes of the security ecosystem in Chapter 3. When *iDefense* or *TippingPoint* buy a vulnerability

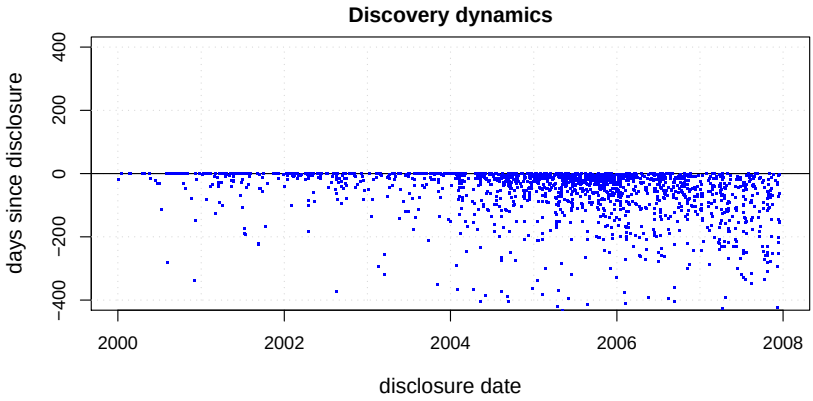

Figure 5.5: *Scatter plot of time of vulnerability discovery*

they record the time of purchase or the time when they notified the vendor of the affected software. Upon the public release, this date can be retrieved from the "disclosure timeline" of their security advisory and serve as an upper bound to estimate $t_{disco}(v)$. Our parser deployed in Phase 1 extracts these dates from the security bulletins of *iDefense* and *TippingPoint*. Using this methodology we determined the time of discovery $t_{disco}(v)$ for a subset $V_{disco} \subset V$ of all vulnerabilities. As of Eq. 3.3 the time of discovery is denoted:

$$t_{disco}(v) \rightarrow time \qquad v \in V_{disco} \subset V$$

Further, as the disclosure of a vulnerability implies its discovery we can state

$$t_{disco}(v) \leq t_{discl}(v) \qquad \forall\ v \in V_{disco} \tag{5.4}$$

Using Eq. 3.7 we can calculate $\Delta t_{disco}(v)$, a minimum estimator for the "pre-disclosure" risk. A vulnerability is discovered before it is reported to OSVDB, the vendor, or sold in the commercial market. Therefore the true "pre-disclosure" risk period is always

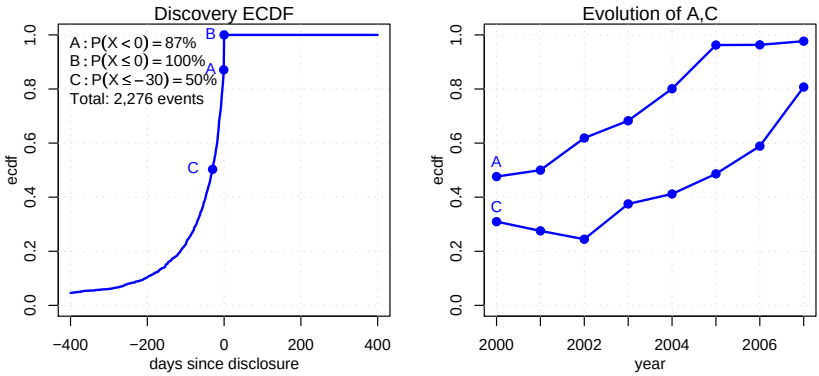

Figure 5.6: *Empirical cumulated distribution of the* discovery time *(left), yearly evolution of selected points in the ecdf (right)*

longer than what we can estimate based on publicly available data. In Fig. 5.6, the values for $x < 0$ show the distribution of the "pre-disclosure" risk from 2000 to 2007. $\mathcal{P}_{\leq}(X \leq x)$ equals 1 for $x \geq 0$ as disclosure implies discovery (Eq. 5.4). In the right Fig. 5.6 we plot the values for (A) $\mathcal{P}_{\leq}(X < 0)$ and (C) $\mathcal{P}_{\leq}(X < -30)$ for each year. The rise of (A) since 2000 simply points out that over time we observe more events with $t_{disco} < t_{discl}$ compared to $t_{disco} \leq t_{discl}$. Further analysis reveals that this is mainly due to the increased share of the "white markets" (which started operation in 2003 and 2005) compared to the OSVDB within our dataset. OSVDB tends to report more vulnerabilities with $t_{disco} = t_{discl}$, while we don't see this for vulnerabilities reported by commercial markets. The course of line (C) $\mathcal{P}_{\leq}(X < -30)$ shows that since 2000 more than 24% of the vulnerabilities were known to insiders more than 30 days before disclosure. In 2007 this share rose to 80% of the vulnerabilities. The course of line (C) is a minimum estimator of the "pre-disclosure" risk. This clearly shows the potential of the abuse of vulnerability information, especially as we have no data on vulnerability discoveries made by

cyber-criminals or traded on the "black market". Applying these results to our model of the processes in the security ecosystem (see Fig. 3.2) we conclude that we are exposed to a considerable "pre-disclosure" risk. Vulnerabilities are systematically known to insiders well before the public learns about it.

5.5 Exploit Availability Dynamics

From several public exploit archives we can find the time of exploit availability for a subset $V_{explo} \subset V$ of all vulnerabilities. For this research we use exploit information found on the well-known public exploit archives *Milw0rm* [86], *Packetstorm* [87], *SecurityVulns* [88], and *Metasploit* [93]. These sources report the date when the exploit was published and a short description of these sources is provided in Appendix A.1. The actual number of exploits available on these exploit archives is bigger than $|V_{explo}|$ as we exclude exploits that cannot be correlated to a given CVE. Further, unpublished vulnerability material in the hands of cyber-criminals will be used for profit, which excludes publicizing an exploit derived from it. As a result, we can only estimate the extent of yet undisclosed exploit information available to them at any time. V_{explo}, based on the content of public exploit archives, is therefore a *minimum estimate* for the true number of exploits available to cyber-criminals at a any given date. As of Eq. 3.4 the time of exploit availability is denoted:

$$t_{explo}(v) \rightarrow time \qquad v \in V_{explo} \subset V$$

Fig. 5.4 shows for what percentage of vulnerabilities we have exploit information available. The scatter plot in Fig. 5.7 shows the distribution of these exploits from 2000 to 2007. We observe that exploits are available *before* and *after* the disclosure of the vulnerability, with an increasing density of exploit availability close to the disclosure day as of 2004. The plot of the cumulated

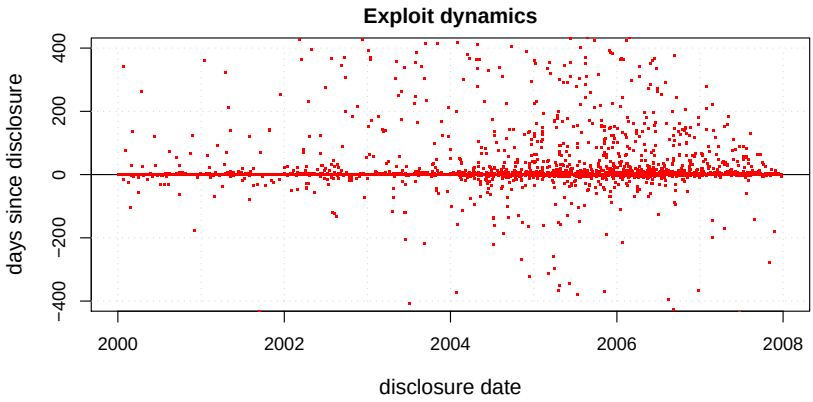

Figure 5.7: *Scatter plot of exploit availability time.*

distribution $\mathcal{P}_{\leq}(X \leq x)$ on the left of Fig. 5.8 quantifies the high dynamics of exploit availability close to the vulnerability disclosure. The sudden rise of $\mathcal{P}_{\leq}(X \leq x)$ from 15% before disclosure to 78% at disclosure from 2000 to 2007 quantifies the so called zero-day exploit[2] phenomena [54]. A zero-day exploit is an exploit that takes advantage of a vulnerability at or before the day the vulnerability is disclosed. In other words, the vendor and the public has zero days to prepare for the security breach. The plot on the right in Fig. 5.8 shows that the zero-day exploit availability is above 70% for the last eight years with the only exception of 58% in 2003.

Zero-day exploit phenomena

We can identify several mechanisms that lead to the very high exploit availability at the time of disclosure. However, we cannot distinguish these mechanisms due to the limited scope and resolution (one calendar day) of publicly available information.

[2] or zero day exploit

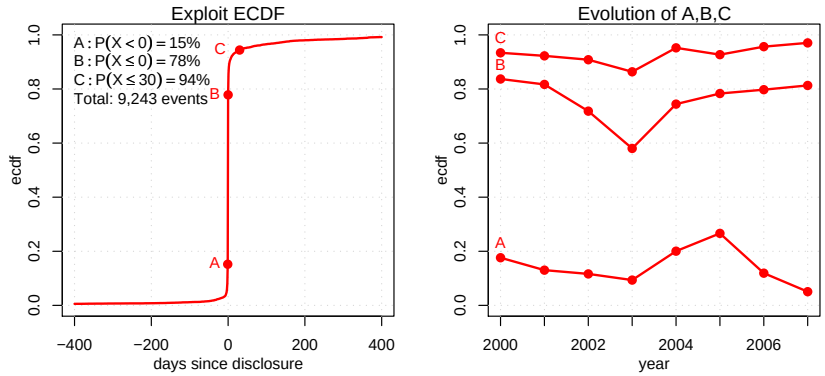

Figure 5.8: *Empirical cumulated distribution of the* exploit availability time *(left), yearly evolution of selected points in the ecdf (right)*

Basically there are two classes of mechanisms depending of whether an exploit is available before the disclosure of the vulnerability or vice versa.

Exploit → Disclosure

Once a gang of cyber-criminals has found or bought a vulnerability they write an exploit. Such an exploit is usually unknown to the public and the product vendor before its release and there is typically limited or no protection to defend against it. However, many vulnerabilities are of a similar nature (mostly Web application based nowadays) and belong to common vulnerability classes such as *XSS*[3], *SQL injection*, and *arbitrarily long input fields* which can be preemptively detected and blocked by intrusion prevention systems (IPS). Cyber-criminals now have two options to take advantage of the exploit: *full scale exploitation* or *stealthy exploitation*.

[3]Cross-site scripting (XSS)

In case of *full scale exploitation* cyber-criminals release the exploit against a large population of targets to take advantage of a greater proportion of unprotected systems. With the higher percentage of compromised systems comes the greater risk of exposure of their activity, which eventually exposes the vulnerability to detection and subsequent disclosure. SIPs and other organizations monitor the (in)security scene, exploit archives, and research malicious activity:

- Anti-virus vendors or providers of managed security services (MSS) capture a sample of the exploit for analysis.
- Honeypots and honeynets capture a sample of the exploit for analysis [94]
- Vendors capture a sample of the exploit through their error reporting mechanisms [95] (usually if the exploit crashes on certain configurations).

These activities lead to the timely disclosure of the underlying vulnerability.

In case of *stealthy exploitation* cyber-criminals use the exploit only against a few, carefully selected high profile targets. Thereby they go at length to prevent detection to extend the time they can profit from the unknown vulnerability [96]. This phenomena is known as "customized malware". However, as described in the "disclosure debate", it is not possible to keep security information secret forever. Eventually, information about the vulnerability spreads and becomes known to a wider audience. When the disclosure of the vulnerability or the release of a patch is imminent, cyber-criminals maximize their return of investment by reverting to *full scale exploitation* of the exploit.

When new exploit material is discovered, it is quickly analyzed and results in the timely publication of a security advisory. Information on high risk exploit material becomes systematically

available to the public with only a short delay. The efficient handling of insecurity information results in a compression of the events for $\Delta t \to 0$ around the disclosure day.

Disclosure → Exploit

Cyber-criminals have also refined their ability to analyze vulnerability information from vulnerability disclosures and reverse-engineering of patches. Recent research demonstrated the potential of automated exploit generation based on a patch [97]. Cyber-criminals quickly create exploits upon the availability of such information.

The combined effect of prior vulnerability knowledge and rapid analysis of disclosed information is readily seen by the increased activity at the disclosure day, and the zero-day exploit availability of close to 80% since 2003. Further, exploit availability reaches 94% 30 days after disclosure. Cyber-criminals systematically take advantage of users failing to install patches quickly, or not having the latest patches installed. We analyze and measure Internet users' discipline of patching their Web browsers in Chapter 7.

5.6 Patch Availability Dynamics

A vendor typically reports the date when a new patch is released together with the patch bulletin or security advisory. To measure the dynamics of patch releases we download, parse, and correlate patch release bulletins of the seven vendors *Adobe, Apache, Apple, Microsoft, Mozilla Foundation, Oracle,* and *RedHat.* We chose these vendors to cover major players of the industry and with respect to the distribution of vulnerabilities among vendors as of Fig. 5.3. Using the release date posted in these vendor bulletins we determine the time of patch availability $t_{patch}(v)$ for

a subset of vulnerabilities $V_{patch} \subset V$. As of Eq. 3.6 the time of patch availability is denoted:

$$t_{patch}(v) \rightarrow time \qquad v \in V_{patch} \subset V$$

Fig. 5.4 shows for what fraction vulnerabilities we have patch information available through the analysis of these seven vendors. The scatter plot in Fig. 5.9 shows the distribution of the availability of these patches from 2000 to 2007. We observe

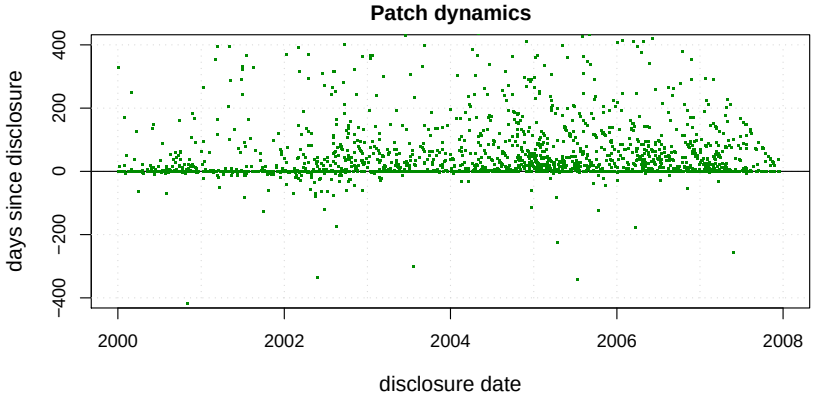

Figure 5.9: *Scatter plot of patch availability time.*

that patches are mostly available *at* or *after* the disclosure of the vulnerability. The plot of the cumulated distribution $\mathcal{P}_{\leq}(X \leq x)$ on the left of Fig. 5.10 quantifies the dynamics of patch availability close to vulnerability disclosure. Essentially, Δt_{patch} reveals the performance of the software industry in providing patches, a measure of the "post-disclosure" risk introduced in Section 3.1.3.

Patch availability 30 days before the time of disclosure is at 2%. There are only few vulnerabilities found for which a patch already exists before the disclosure. The sudden rise of $\mathcal{P}_{\leq}(X \leq x)$ from 6% one day before disclosure to 43% at disclosure from 2000

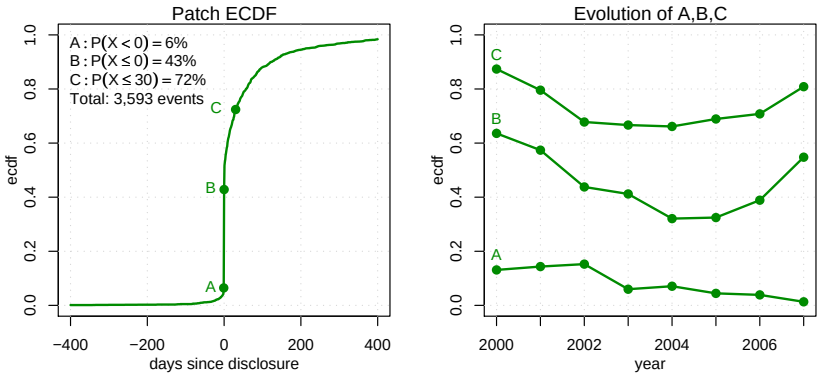

Figure 5.10: *Empirical cumulated distribution of the* patch availability time *(left), yearly evolution of selected points in the ecdf (right)*

to 2007 quantifies what we call the *zero-day patch* phenomenon. The fraction of zero-day patches can be interpreted as a measure of the *responsible disclosure process*. Before a patch is ready for publication the vendor needs time to analyze the vulnerability, develop, test, document, and finally release the patch. Typically, a vendor is unable to release a patch within 24h of vulnerability discovery. Thus, to achieve a zero-day patch the vendor needs early notification of the vulnerability, typically through the responsible disclosure process, which includes contributions by the "white market" analyzed in the next section. Further, we discuss the zero-day patch phenomenon in detail in Chapter 6. The rise of $\mathcal{P}_{\leq}(X \leq x)$ for $x > 0$ measures how vendors react to vulnerability disclosures. Patch availability increases from 46% at disclosure to 72% at 30 days after the disclosure (equalling 28% unpached vulnerabilities 30 days after disclosure). This is a low number compared to the exploit availability of 94% 30 days after disclosure. 13% of the vulnerabilites are still unpatched 90 days after the disclosure.

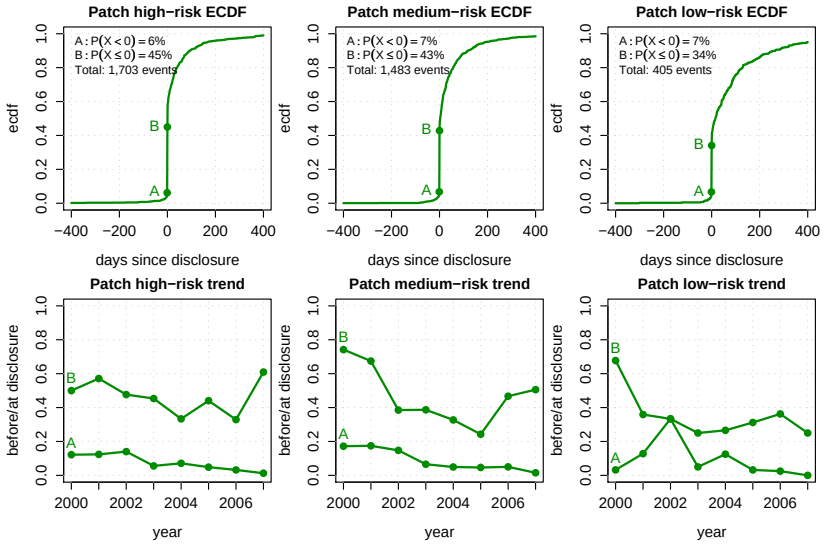

Figure 5.11: *Patch availability by risk class "high", "medium", and "low"*

To determine how the risk of a vulnerability affects the patch performance we separately analyze the data for the three risk classes "high", "medium", and "low". In Fig. 5.11 we show for each risk class the aggregated distribution of patch availability on top, and the course of $\mathcal{P}_{\leq}(X < 0)$ (A) and $\mathcal{P}_{\leq}(X \leq 0)$ (B) for each year below. This analysis indicates that the patch performance of "low" risk vulnerabilities consistently lags behind the performance of "high" and "medium" risk vulnerabilities. Aggregated over the years the patch availability *before* disclosure (A) is comparable for all risk classes with 6% to 7%. However, *at* disclosure (B) the performance for "high" (45%) and "medium" (43%) risks exceeds the "low" (34%) risk performance. While "high" and "medium" risk patch availability evolve almost alike *after* disclosure, the distribution of "low" risk patch availability stays considerably behind. From the yearly evolution shown in

the lower part of Fig. 5.11 we see that “high” and “medium” risk patch availability improved in the last 2-3 years while the “low” risk patch availability mainly decreased since 2000. We infer from these observations that the risk class of a vulnerability marginally affects the patch release performance in the sense that patches for “high” and “medium” risk vulnerabilities are prioritized against patches for “low” risk vulnerabilities. If technological complexity to fix a vulnerability would be the dominant parameter to determine patch performance, then our measurements would lead to the conclusion that “low” risk vulnerabilities are generally more complex to fix than “high” or “medium” risk vulnerabilities. We consider this unlikely. We rather assume that work flow processes and priorization (and with it incentives) are at least as important as technical complexity to determine patch performance.

Note that the discovery of a vulnerability by the vendor itself is also considered as responsible disclosure. An appropriately motivated employee discovering a vulnerability could also choose to offer this information to cyber-criminals instead. Applying these results to our model of the processes in the security ecosystem (Fig. 3.2), we conclude that between 6% and 43% of the vulnerabilites of the analyzed vendors followed the process Path (D) or Path (E). An analysis of the contributions of the “white market” helps to estimate the prevalence of Path (E).

5.7 Prevalence of the “white market”

To estimate the prevalence of Path (E) in the security ecosystem model we look at the two main vulnerability purchase programs in the “white market”, VCP and ZDI, as introduced in Chapter 3 on Page 35. Together, VCP and ZDI published 793 vulnerabilities affecting 192 different vendors since their start in March 2003 to December 2007. In the same period a total of 8,111 vulnerabilities were published for the same group of 192 vendors, including

the 793 bought by VCP and ZDI. We normalize the number of

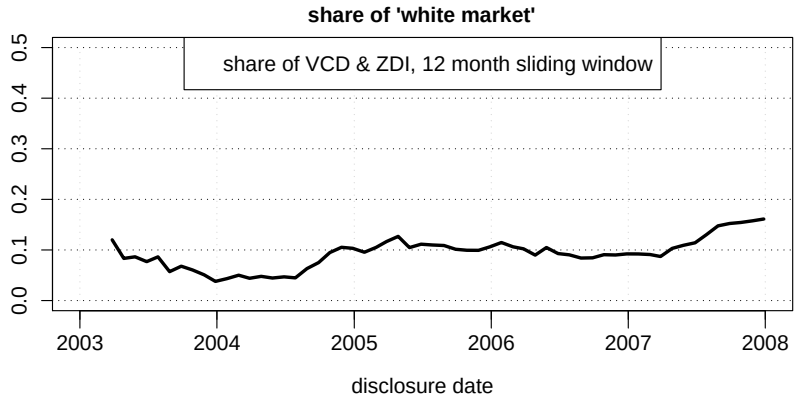

Figure 5.12: *Share of commercial vulnerability purchase programs in 12 month moving window.*

"white market" vulnerabilities with respect to the total number of vulnerabilities disclosed for the group of affected vendors in the same period to estimate the prevalence of the "white market". Using a sliding window approach with window size $t_{win} = 12\ month$ we calculate the share of the "white market" within the group of vendors for which VCP and ZDI bought vulnerabilities, shown in Fig. 5.12. We observe an almost constant share of about 10% of these commercial programs since the end of 2004 and a rise to over 15% starting in 2007. These numbers shed a first light on the extent that "white markets" contribute to the vulnerability ecosystem. Fig. 5.12 shows the prevalence of Path (E), which at the same time provides a *minimum estimate* of the number of vulnerabilities *not* discovered by the vendors themselves. For example, between March 2003 and December 2007 7.5% of the vulnerabilities affecting Microsoft and Apple were processed by either VCD or ZDI, while other vendors achieved higher shares.

5.8 (In)Security Dynamics

So far we have investigated the individual distributions of the patch- and exploit-times. An interesting aspect of our analysis is the direct comparison of these two distributions. Fig. 5.13 depicts the cumulated distribution of the exploit-, and patch-time for direct comparison and Fig. 5.14 and Fig. 5.15 show the trend over the last five years.

5.8.1 The Gap of Insecurity

We plot the cumulated distribution $\mathcal{P}_{\leq}(X \leq x)$ of $\Delta t_{patch}(v)$ for all vulnerabilities $v \in V_{patch}$ of the seven vendors listed in Section 5.6 together with the cumulated distribution $\mathcal{P}_{\leq}(X \leq x)$ of $\Delta t_{explo}(v)$ for all $v \in V_{explo}$.

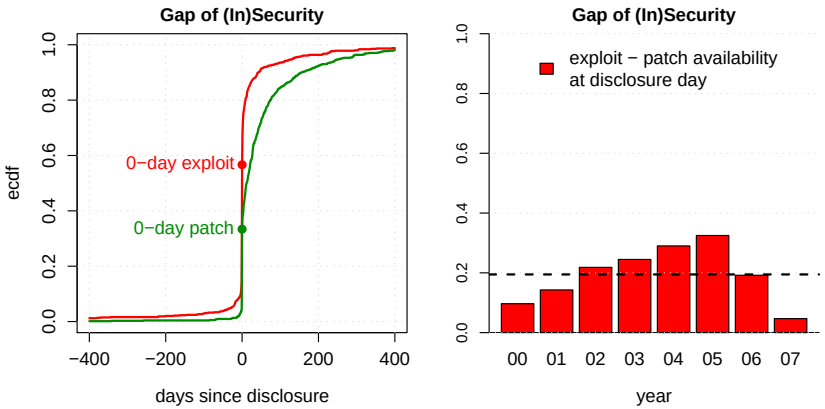

Figure 5.13: *Direct comparison of exploit availability vs. patch availability*

Through vendor Web sites we have systematic access to *all patches* published by a given vendor and V_{patch} contains *all patches* published by these seven vendors. On the other hand, not

all exploits are made available on public exploit archives. Profit-motivated cyber-criminals release their exploits against targets and have no incentive to put their material on public exploit archives. Eventually, some of the exploits used exclusively by cyber-criminals will make their way into exploit archives (as an exploit, proof of concept, test for patch). However, these postings happen with a delay. On the other hand, cyber-criminals monitor exploit archives and quickly enhance their repository of malware should they find material previously unknown to them. We therefore conclude that $|V_{explo}|$ is a *minimum estimate* of the true number of exploits available in the wild at any time. Further, the distribution of $\Delta t_{explo}(v)$ is a *lower estimate* of the exploit availability. True exploit availability is always larger (=faster). On the right Fig. 5.13 we plot the difference of *exploit availability* and *patch availability* at $x = 0$ for each year since 2000. We find

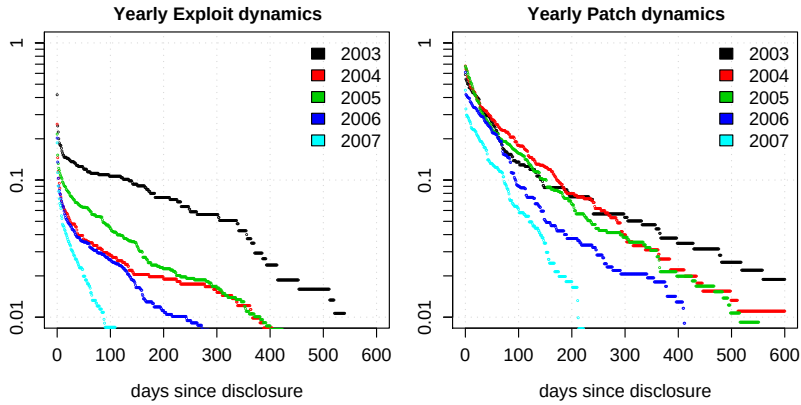

Figure 5.14: *Evolution of yearly exploit and patch availability 2003 to 2007.*

that exploit-availability continuously exceeds patch-availability for the full range ±400 days around the disclosure. Further, exploit availability consistently exceeds patch availability in every

year since 2000. This gap, which quantifies the difference between exploit- and patch-availability, is an indicator of the risk exposure and its development over time. This systematic gap also stresses the importance of the availability of independent and timely security information.

5.8.2 Evolution and Event Compression at Zero-Day

To analyze the evolution of the dynamics of exploit- and patch availability we first plot the respective distributions for the last five years for $t \in [0,600]$ days after disclosure. We use the complementary cumulative distribution function (CCDF) $\mathcal{P}_{>}(X > x)$ as of [89].

$$\mathcal{P}_{>}(X > x) = \mathcal{P}_{>}(x) = 1 - \mathcal{P}_{\leq}(x) \tag{5.5}$$

of exploit- and patch-availability. In Fig. 5.14 we plot the

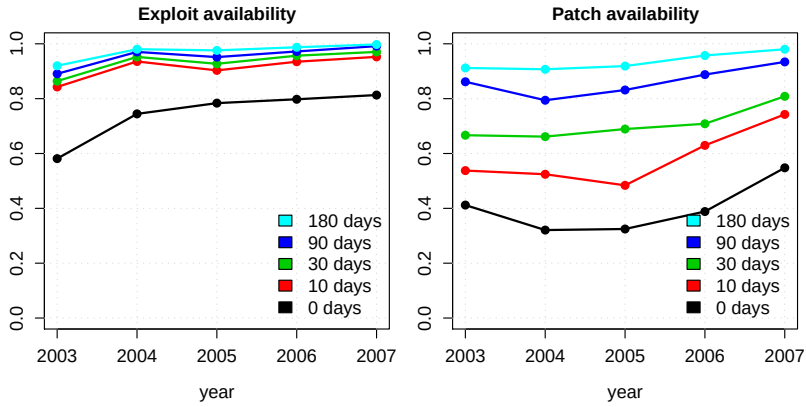

Figure 5.15: *Evolution of yearly exploit and patch availability 2003 to 2007.*

$\mathcal{P}_{>}(X > x)$ for every year since 2003 on a $log - linear$ scale and

in Fig. 5.15 we plot selected values of the *ecdf* at 0, 10, 30, 90 and 180 days to visualize the evolution of the dynamics of exploit availability (left) and patch availability (right). Note that high values on the $y-axis$ in Fig. 5.14 translate to a low availability of exploits or patches as of Eq. 5.5. Generally, both exploit and patch availability increased in the last five years.

With the exception of 2005, exploit availability increased steadily since 2003 and we observe a higher rise close to the disclosure day. Exploit availability 30 days after disclosure continuously exceeds 90% since 2004. We observe high exploit dynamics within 10 days of disclosure, thereafter exploit availability rises only very slowly, although on a high level close to 100%. We attribute this observation to the following causes:

- Exploits already known to cyber-criminals *before* the public disclosure of the vulnerability.
- Increased capability to generate exploits either through reverse-engineering of patches or based on disclosed vulnerability information.
- Automated attack tools for Web application vulnerabilities that can actually *discover and exploit* a vulnerability. It is only afterwards that the consultant/user of the tool realizes that the vulnerability exists - and then informs them that they need to fix it.

We cannot distinguish these causes based on our data, we measure the aggregate effect. Note that our data is a minimum estimate of the true availability of exploits.

On the other hand, also patch availability increases almost steadily over the last years, although starting from a lower level than exploit availability. Closer to the disclosure, patch availability first dipped around 2005 and then caught up in the last three years. Again, patch availability is always lower

than exploit availability at any day. Patch availability 90 days after disclosure does not surpass exploit availability 10 days after disclosure. We attribute patch availability performance to two different processes:

- *Patch release at zero-day* The release of a patch at the same day as the public disclosure of the vulnerability implies the vendor had early notification of the vulnerability ("responsible disclosure"). A vendor is typically not able to analyze vulnerability information, develop, test, and release a patch in less than a day. However, whether a vendor receives early notification from vulnerability discoverers is only partially under control of the vendor. This is to a high degree an exogenous factor the vendor can only control in the long term by establishing a trust relationship with the security community. We discuss zero-day patches in detail in the next chapter.

- *Patch release after disclosure* The time needed to release a patch upon knowing the vulnerability is under control of the vendor, a endogenous factor. Here we measure what a vendor *can do*, and what he is *willing to do* given technological complexity to fix the software, and economic incentives or constraints.

We believe that a good relationship with the security community can provide a higher share of early notifications of vulnerabilities which benefits a vendor in the following ways:

- Within "responsible disclosure" the vendor has more control of the time available to develop and release a patch than under the pressure of an already published vulnerability. This will typically result in a more efficient allocation and use of available resources of the vendor.

- A higher share of zero-day patches will be perceived as a better service to the customer.

The observed trend to increased patch availability *at* and *after* the public disclosure indicates that the processes involved to release patches (technological, economic, incentives) have not yet reached saturation. Continued measurements using the methodologies presented in this chapter should be able to identify the limits of such processes at macroscopic scale.

5.9 Summary

We examined the availability of exploits and patches since 2000 in relation to the disclosure of the vulnerability. We found very high dynamics at the day of disclosure, for both exploits and patches, and explained the processes leading to this observation. While we have complete data on patch availability for the vendors examined, we can only give a *minimum estimate* for the corresponding exploit availability. We found that exploit availability consistently exceeds patch availability since 2000. We measured the extent of this effect and call it the *gap of insecurity*. This gap of insecurity clearly demonstrates the need of third party protection (e.g., anti-virus, intrusion protection systems, filtering devices), as vendors are in large parts unable to protect their customers through the release of patches. The complexity and the delay of installing patches paired with the fact that we can only provide an minimum estimate for exploit availability stresses the need for third party protection *and* timely availability of security information to the public. Our measurements confirm the effective monitoring of the (in)security scene by SIPs and stress the importance to have access to unrestricted, timely and independent security information. Our measurement methods are based entirely on publicly available information and provide a useful tool to measure the state of the security ecosystem and its evolution over time. The presented data on the zero-day patch

availability suggests that the concept of responsible disclosure is effective.

Chapter 6

Patch Performance Metric

6.1 Introduction

The ever continuing discovery of new vulnerabilities and exploits drives the security risks we are exposed to. While the availability of a patch at the same time as the discovery of a new vulnerability could greatly reduce or eliminate the risk, the time required for the patch development and testing render this scenario impossible. In practice, vendors publish patches as soon as these are available or they publish them on a predefined schedule to ease the planning of patch implementation. To better understand the security ecosystem, we measure and compare the performance of the vulnerability handling and patch development process of two major software vendors, namely Microsoft and Apple. We introduce the *zero-day patch share* and the *patch backlog* as new metrics to measure and compare vendors' performance in releasing security patches.

6.2 Methodology

6.2.1 Selection of Vendors

We choose Microsoft and Apple for this case study of vendor patch performance. Both Microsoft and Apple have developed their own operating systems "Windows" and "Mac OS", each with a rich set of specific applications for private and professional users. Both vendors are well known and established as major software manufactures in the market. Microsoft and Apple both occupy high ranks in Fig. 5.3 on Page 87, the list of the top-10 most vulnerable vendors over the last years. To analyze the patch performance of Microsoft and Apple, we look at the period from January 2002 to December 2007. For this case study. we use a subset of the information gathered in the analysis of the previous chapters, specifically we use the disclosure time $t_{discl}(v)$ and the patch time $t_{patch}(v)$ of vulnerabilities of these vendors.

6.2.2 Selection of Vulnerabilities

To measure the performance of a vendors' patching process we are only interested in vulnerabilities the *specific vendor felt responsible for*. This criterion excludes vulnerabilities of third-party tools, software, and libraries that might be included in, or run on Microsoft or Apple products. Therefore, we limit this analysis to vulnerabilities for which the particular vendor eventually released a patch. Releasing a patch implies that the vendor felt responsible for the vulnerability and that the severity justified doing so. Every attempt to broaden the number of vulnerabilities would introduce a bias (a) when deciding if a certain vulnerability should be attributed to a vendor; and (b) if the severity/risk of the vulnerability justifies inclusion into the analysis. With the release of a patch for a vulnerability the vendor positively and unmistakenly takes responsibility for it,

with respect to the origin of the vulnerability and the security impact.

6.2.3 Risk Level of Vulnerabilities

We use the national vulnerability database (NVD), which is vendor independent, to determine the risk rating of the patched vulnerabilities. Further, we only include high-, and medium-risk vulnerabilities for this analysis. This restriction is introduced because a vendor could always argue that patching a low risk vulnerability is not time critical, and therefore delayed. This stance is much less likely to be accepted by users and customers in case of higher risk vulnerabilities. Tables 6.1 and 6.2 list the number of vulnerabilities per vendor and their risk rating.

Microsoft

Year	High	Med	Low	Total
2002	100	44	1	145
2003	58	22	1	81
2004	58	28	3	89
2005	50	26	4	80
2006	72	82	11	165
2007	87	31	0	118
Total	425	233	20	678

Table 6.1: *Out of a total of 678 patches, Microsoft released 658 patches for high and medium risk vulnerabilities from 2002 to 2007.*

Apple				
Year	High	Med	Low	Total
2002	33	18	2	53
2003	34	31	3	68
2004	63	61	9	133
2005	58	72	32	162
2006	62	92	20	174
2007	115	99	6	220
Total	365	373	72	810

Table 6.2: *Out of a total of 810 patches, Apple released 738 patches for high and medium risk vulnerabilities from 2002 to 2007.*

We denote $V_{MS} \in V$ and $V_{APPLE} \in V$ as the set of medium and high risk vulnerabilities Microsoft and Apple patched from 2002 to 2007 with $|V_{MS}| = 658$ and $|V_{APPLE}| = 738$.

6.2.4 Zero-Day Patch Metric

Based on the definition of the lifecycle of a vulnerability in Section 3.1 and the related exposure phases, we denote a *zero-day patch* to be any patch where the vulnerability is disclosed at the same day the patch is released. A vendor always needs time to analyze a vulnerability and develop, test, document, and finally release the patch. Thus, to ever achieve a zero-day "post-disclosure" risk period, the vendor imperatively needs prior information about the vulnerability. This is typically achieved though the "responsible disclosure" process described in Section 3.4.2 of Chapter 3. Hence, the zero-day patch share is a viable estimator of the responsible vulnerability disclosure process for a specific vendor. Note that the discovery of a vulnerability by the vendor itself is also considered as responsible

disclosure[1]. This indicator rewards vendors that cooperate well with the security community, e.g., by setting up processes and policies that foster coordinated disclosures. To achieve a better zero-day patch share, a vendor could potentially only release patches for which he achieves a zero delay, omitting the development or release of other patches. We consider this scenario as highly unlikely. In a competitive market a vendor cannot afford to ignore high or medium risk vulnerabilities at large, as shown in the discussion of the security ecosystem in Chapter 3. If ignoring vulnerabilities would be a viable option, then not releasing patches at all would be an viable option too, which is not what we whiteness in todays environment.

We generalize the zero-day patch metric and also measure the share of patches available within $\Delta d \in \{0, 30, 90, 180\}$ days of disclosure using a sliding window of size $w = 360$ days for any day t in the period of January 2002 to December 2007. This window size is chosen to minimize seasonality effects during a calendar year. For all vulnerabilities $v \in V^{win}$ *disclosed* within the sliding window

$$V^{win}(t,w) \rightarrow \{v \in V_{vendor} | (t-w) < t_{discl}(v) \leq (t)\} \tag{6.1}$$

we determine the share $\mathcal{P}_{\leq}(X \leq \Delta d) = ecfd_{patch}(\Delta d)$ of patches the specific *vendor released* within Δd days of disclosure using Eq. 5.3. We plot this metric in Fig. 6.1 for Microsoft and Fig. 6.2 for Apple. As we plot distinct points of $\mathcal{P}_{\leq}(X \leq \Delta d)$ at $\Delta d \in \{0, 30, 90, 180\}$ days, the following relation holds:

$$\mathcal{P}_{\leq}(0) \leq \mathcal{P}_{\leq}(30) \leq \mathcal{P}_{\leq}(90) \leq \mathcal{P}_{\leq}(180) \tag{6.2}$$

A vulnerability patched at day 0 is included in the curve showing vulnerabilities patched no later than 30 days after the disclosure, etc.

[1]The employee discovering a vulnerability could also choose to offer this information to cyber-criminals

6.2.5 Patch Backlog Metric

To complement the zero-day patch metric we introduce the *patch backlog metric*. While the zero-day patch share measures the percentage of vulnerabilities a vendor releases within a given delay Δd from their disclosure, the patch backlog metric measures the absolute number of unpatched, but publicly known, vulnerabilities at a given date. For every vulnerability v in V_{MS} and V_{APPLE}, we know the disclosure date and the patch date. Starting with a patch backlog of $y = 0$ on 2002-01-01, we increment y for every disclosure and decrement y for every patch release according to

$$y \to y+1 \quad \text{at } t_{discl}(v) \text{ for all } v \in V_{vendor} \tag{6.3}$$

$$y \to y-1 \quad \text{at } t_{patch}(v) \text{ for all } v \in V_{vendor} \tag{6.4}$$

By definition, y starts at 0 on 2002-01-01 and ends at 0 on 2007-12-31 as we only look at vulnerabilities the respective vendor patched within our observation period. We plot this metric in Fig. 6.4 for Microsoft and Apple.

6.3 Case Study Microsoft vs. Apple

6.3.1 Zero-Day Patch

We plot the zero-day patch metric for all high-, and medium-risk vulnerabilities $v \in V_{MS}$ and $v \in V_{MS}$ of Microsoft and Apple in Fig. 6.1 and Fig. 6.2. The lowest curve shows the share of patches that were available at the exact day of the vulnerability disclosure, the zero-day patches. The curves on top of the zero-day patch share show the total share of patches available after $\Delta d \in \{30, 90, 180\}$ days of the disclosure time[2]. In Table 6.4 we

[2]We had data before 2002 to ensure the respective sliding windows are filled at 2002-01-01

summarize the average share for these delays Δt for both vendors. Within the six-year observation period we find a high variation of the zero-day patch share for both vendors, with Microsoft achieving between 30% and 91% and Apple between 0% and 68%. Before mid 2003, Apple's zero-day patch share is 0, while they still suffered from vulnerabilities as can be seen in the 30, 90, 180-day shares during that period. Remarkably, both vendors do not achieve 100% patched vulnerabilities within 180 day of disclosure for extended periods. We find the average share of unpatched vulnerabilities 180 days after disclosure to be 6% for Microsoft and about 12% for Apple. A comparison with the content of Tables 6.1 and 6.2 reveals that the zero-day patch performance does not correlate with the absolute number of vulnerabilities disclosed in a given year. For example, between 2005 and 2006 the absolute number of patches released by Microsoft doubled while the zero-day patch share increased. In the last 6 years, Apple's zero-day patch share was higher than Microsofts' for only about one year at the end of 2006. As previously explained,

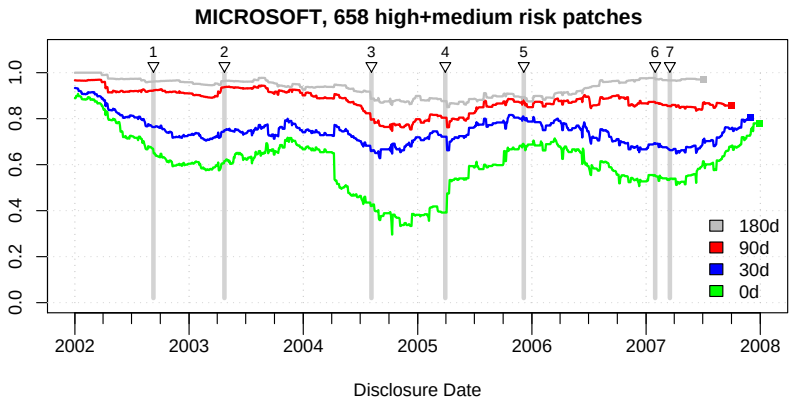

Figure 6.1: *Share of Microsoft patches available within 0, 30, 60, and 180 days of vulnerability disclosure.*

the release of a zero-day patch implies the vendor had advance

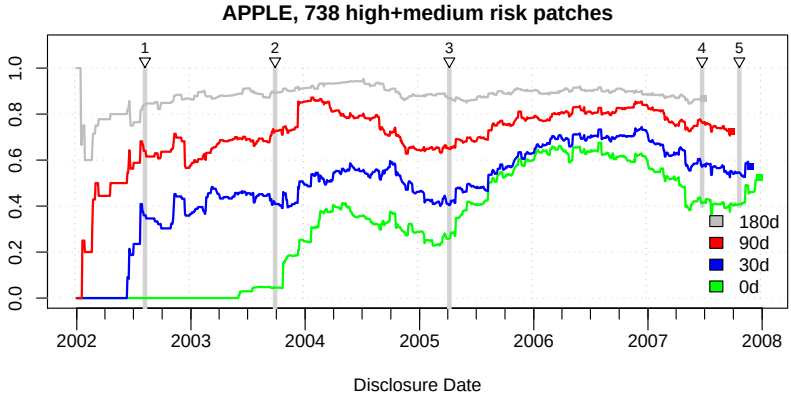

Figure 6.2: *Share of Apple patches available within 0, 30, 60, and 180 days of vulnerability disclosure.*

notification of the vulnerability, which in turn implies that the discoverer of the vulnerability is willing to report the finding to the specific vendor. In our security ecosystem model, the zero-day patch rate measures the prevalence of process Path (D) and Path (E) as of Fig. 3.2 on Page 34. Whether one chooses to report a vulnerability to a specific vendor strongly depends on how a vendor is known to treat the discoverer.

We believe that our measurements reflect how Microsoft and Apple are perceived by the security community since 2000. Fig. 6.1 and Fig. 6.2 indicate that Microsoft started earlier building a good relationship with vulnerability discoverers than Apple, which conforms to the observation in [75]. In January 2000 Microsoft began a new policy regarding acknowledgments in security bulletins [98] and extensive material on how to talk security with Microsoft is available and documented on their Web site. On the other hand, Apple still lacks such documentation and statements. Security researchers complain about this [99], and revert to *full disclosure* instead. From Table 6.3 we see that on

average, the vendor had prior knowledge (*responsible disclosure*) in 61% of the cases for Microsoft and 32% for Apple.

Vendor	**min**	**max**	**mean**	**sdev**
Microsoft	30%	91%	61%	12%
Apple	0%	68%	32%	23%

Table 6.3: *Zero-day patch share from 2002 to 2007*

Vendor	**0-day**	**30-day**	**90-day**	**180-day**
Microsoft	61%	75%	88%	94%
Apple	32%	49%	71%	88%

Table 6.4: *Average Δd-day patch share from 2002 to 2007*

Competition of resources

With numbered vertical lines in Fig. 6.1 and Fig. 6.2 we visualize the release date of major software upgrades or new versions of the operating system of Microsoft (Table 6.5) and Apple (Table 6.6). We find that the patch development performance shows a correlation with major software releases of both vendors. It appears that the parallel development of Windows XP SP2 (released August 2004) and Windows Server 2003 SP1 (released March 2005) has absorbed considerable resources at Microsoft at the cost of patch development in the month before release. Similarly the development of Mac OS X 10.3 Panther (released October 2003) and Mac OS X 10.4 Tiger (released April 2005) appears to have absorbed considerable resources at Apple. In general, a major software release seems to have positive impact

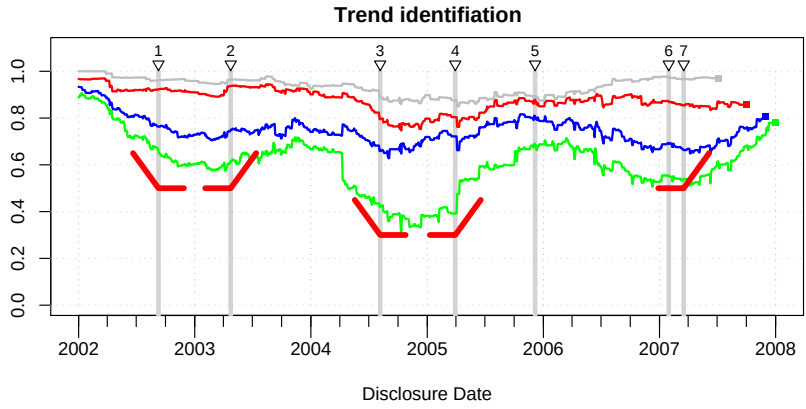

Figure 6.3: *Illustration of trend reversal after major software releases.*

on the zero-day patch rate in the following months. The zero-day patch share either increases, or stops the decrease prevalent in the month before the release of the software package. In Fig. 6.3 we exemplify this on some events where the zero-day patch share either *increased*, or *stopped decreasing* after the release of a major software package.

ID	Date	Event
1	2002-09-09	WinXP SP1
2	2003-04-24	WinSrv 2003
3	2004-08-06	WinXP SP2
4	2005-03-30	WinSrv 2003 SP1
5	2005-12-05	WinSrv 2003 R2
6	2007-01-30	WinVista
7	2007-03-13	WinSrv 2003 SP2

Table 6.5: *Major software releases by Microsoft*

ID	Date	Event
1	2002-08-13	OS X 10.2 Jaguar
2	2003-10-03	OS X 10.3 Panther
3	2005-04-12	OS X 10.4 Tiger
4	2007-06-29	iPhone
5	2007-10-26	OS X 10.5 Leopard

Table 6.6: *Major software releases by Apple.*

6.3.2 Overdue Patches

In Fig. 6.4 we plot the cumulated number of unpatched vulnerabilities for Microsoft and Apple in the period of January 2002 to December 2007. Timely release of patches for disclosed vulnerabilities will lead to low values of the cumulated number of unpached vulnerabilities. As we only include vulnerabilities in this analysis for which a patch was available no later than December 2007, the vendors' curves start and end at zero. We find the total number of unpatched vulnerabilities at any day within the observation period varies between 0 and 22 for Microsoft and between 0 and 55 for Apple. The mean number of concurrent unpatched vulnerabilities is 11.4 for Microsoft and 24.7 for Apple. On average, Microsoft succeeds in keeping the number of unpatched vulnerabilities below 20.. On the other hand, Apple seems unable to stabilize the number of unpatched vulnerabilities and we observe a steady increase in recent years for Apple. It seems that Apple's security processes and resources cannot cope with the side-effects of the increased popularity of their products. A vendor could potentially reduce the absolute number of patches by prematurely ending the support for older

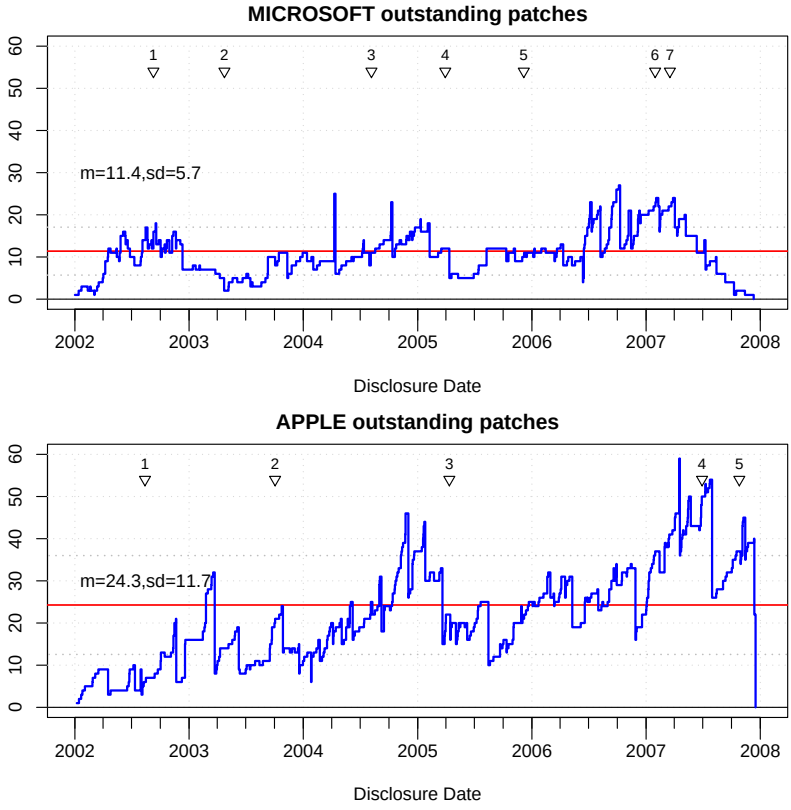

Figure 6.4: *Number of overdue patches for Microsoft and Apple at any day from 2002 to 2007.*

software versions in order to look better on this metric. However, customers and users of still prevalent older versions would not appreciate such a stance. Economic and customer pressure forces vendors to patch still popular versions. Microsoft, for example, had to extend the support of the still popular Windows XP operating system contrary to its original plans [100].

Starting in early 2004, both vendors show an continued increase of the number of unpatched vulnerabilities just to fall remarkably in the first months of 2005. In the same period both vendors show in parallel a significant decrease of the zero-day patch share.

The parallel drop in patching performance of both vendors in the period 2004 to 2005 may be explained as the effect of vendor independent, exogenous factors. Possible reasons (among others) are the availability of new hacker/security tools and techniques (e.g., like fuzzing) or changes in the methodology of software development processes (e.g., better security testing will reveal more vulnerabilities).

6.4 Summary

We evaluated the patch development process of Microsoft and Apple using publicly available vulnerability data from 2002 to 2007. We introduced the *zero-day patch share* and the *patch backlog* as new metrics to measure and compare different vendors patch release performance. These metrics do not allow the ranking of the security of the products of one vendor or another, but they do give insight into the security processes of the vendors. With the zero-day patch share we empirically measure the prevalence of the important responsible disclosure process, a cornerstone in the security ecosystem. We find that *responsible disclosure* in general works, and that the prevalence depends on the specific vendor. Our metrics are unbiased, as they are entirely based on publicly available data which is vendor independent. The disclosure date is collected from a set of independent and competing SIPs and the patch date is collected from a vendors security bulletins. Upon the release of a patch, the vendor cannot manipulate the release date (e.g., predate the patch) without detection. Further, not

releasing a patch at all is no viable option for a vendor in a competitive market. Using these metrics we measured the processes of major vendors how they handle security information and develop patches. We found a correlation between major software releases and patch development performance that we contribute to a competition of resources withing a vendor. Actual measurements of vendor performance is an necessary first step in understanding the processes in the security ecosystem. Using the zero-day patch metric we can measure the prevalence of responsible disclosure, which reflects a kind of equilibrium between the processes in the security ecosystem. Extending these measurements to more vendors and tracking it for future years helps to understand the level of protection we can realistically expect from software vendors in general, or from a specific vendor. For example, independent patch performance information, if available, could become a parameter of new software platform or product purchase decisions. A future comparison between vendors can potentially identify processes and behaviors that lead to better security.

Chapter 7

Browser Patch Update Dynamics

In the previous chapters we discussed and analyzed the dynamics of the vulnerability lifecycle events *discovery*, *exploit availability*, *disclosure*, and *patch availability* introduced in Chapter 3. The availability of a patch does not protect any system until users of the software eventually install the patch. The delay to patch installation measures the "post-patch" risk which cannot be represented as a distinct value for a given vulnerability, rather it is a distribution describing how promptly a given population of software users installs a new patch. In this chapter we analyze *patch installation dynamics*. Corporate users in managed environments not only benefit from various protection mechanisms not available to ordinary home users, patching of their systems is centrally managed by their organization. Patch implementation dynamics can be measured in such environments, but the data is proprietary and not publicly available. On the other hand, to this day no methodology or empirical data exists on the end-users' patch implementation dynamics on global scale. We present a new methodology to do so and measure the patch

implementation dynamics of the most frequently used client application in the Internet, the *Web browser*, which in recent years has become the primary infection vector for vulnerable hosts. Failure to apply patches promptly or missing them entirely is a recipe for disaster in today's hostile Internet, exposing the host to infection and possibly subsequent data disclosure or loss. In fact, a visit to a single malicious Web site with an unpatched browser is enough to get compromised. In this chapter we present a new methodology to measure the patch level of Web browsers, and thereby the "post-patch" risk as defined in Section 3.1.3, without the need to access the computer of the end-user or to install any kind of monitoring software on it.

7.1 Methodology

In order to measure the distribution and evolution of the patch level of Web browser populations on global scale, we exploit the information in the *HTTP user-agent string* submitted with every Web page request. This information is readily available in the log files of most Web servers. For this research we analyze anonymized log files of Google's search and application servers, covering 75% of the world's Internet users for more than a year. We first discuss the dynamics of major version migrations of the four most popular Web browsers, namely Microsoft's *Internet Explorer* (IE), Mozilla's *Firefox* (FF), Apple's *Safari* (SF), and Opera's *Opera* (OP). We then focus our analysis on the Web browsers Firefox and Opera as these browsers provide detailed minor version information in the user-agent string and are freely available for multiple operating systems. This analysis, combined with a catalogue of known vulnerabilities and subsequent security patches associated with a particular update, enables us to estimate the lower bound of the number of Web browsers in use repeatedly failing to apply patches, many of which fixed

built-in browser vulnerabilities. Armed with the results of our measurements we discuss software update mechanisms and draw conclusions with respect on how to better protect the global user population.

A Web browser sends the user-agent string in the HTTP protocol header with every request for a Web page. This string contains the *type* and *version* of the browser and the type of *operating system* the browser runs on [101–103]. To measure the number of unique browser installations active on a given day we need a way to reliably remove duplicates resulting from multiple visits to a Web site. Relying on the client's IP address is not sufficient as a large user base surfing behind proxies is seen through a single IP address only. For our study we rely on Google's PREF cookie to eliminate duplicate visits by the same browser. We ignore the small fraction of browsers that disabled cookies due to restrictive user settings. We also ignore the small possibility of cookie id collisions and the effect of users deleting cookies manually. While Google offers its Web search also anonymously without requiring cookies, by default users allow cookies to remember their preferences. To protect users' privacy, Google was the first search engine to publicly commit to anonymize cookies and IP addresses in logs after 18 months and to shorten the cookie expiration time to two years. Some browsers and browser extensions allow the user to modify the information in the user-agent string and there are Web spider tools that impersonate a widespread browser for compatibility. There are also some proxies that change the user-agent string. Based on the observed dynamics of Web browser update installation we expect this effect to be small.

7.1.1 Data Mining

We analyze Google's Web server log data from January 2007 to July 2008 with typically three samples per week, namely

Monday, Wednesday and Saturday (Pacific time zone GMT-8). Starting Oct 10th, 2007 we use daily samples. The log-parser was implemented as a MapReduce [104] that runs on hundreds of machines and processed anonymized Google Web server logs during our observation period. For every day sampled, it identifies and counts known user-agent strings for Firefox, Internet Explorer, Safari, and Opera. In sum, all other user-agent strings found account for less than 1% of the share and are mostly attributed to non mainstream browsers, mobile devices, automated tools, and proxies. From the identified user-agent strings of the four major Web browsers we derive the major and minor version of the respective Web browser type, which provides us the information of the patch level on a per day basis. This information is then correlated with the release date of Web browser patches to estimate the "post-patch" risk exposure.

7.2 Major Version Migration Dynamics

7.2.1 Major Version Migration

In mid June 2008, the most commonly encountered browser technologies used to navigate the Internet were Internet Explorer (78%), Firefox (16%), Safari (3%), and Opera (1%) according to TheCounter.com [105]. The combined usage share of these four browsers was 98.6%, dominated by Internet Explorer and Firefox as can be seen in Table 7.1. To assess the dynamics of Web browser major and minor updates we first measure the transition between the most recent major versions within our observation period. The migration to the next major version of a browser usually requires a manual installation. Minor version updates typically are highly automated, depending on the type of browser. For comparison we measure major version migrations of Internet Explorer, Firefox, Safari, and Opera. Mozilla released FF2 in October 2006. There were 14 updates

of minor versions for FF2 and three updates for FF1.5 released in our observation period from January 2007 to June 2008. In the same period, Apple released the new major version 3.x of its Safari browser. However, Microsoft (IE7) and Opera (OP9) released their most recent major versions before our observation started, and Google's Chrome browser and Mozilla's Firefox 3.0 were released thereafter.

Share of browser type		
Browser	Share	Mio.
IE	78.3%	1,103
FF	16.1%	227
SF	3.4%	48
OP	0.8%	11
Total	**98.6%**	**1,389**

Table 7.1: *Share of Web browsers by type according to TheCounter.com averaged over Feb 1st to June 18th, 2008. The absolute worldwide user counts were derived from [1] as 1,408 billion users.*

In Fig. 7.1 and Fig. 7.2 we plot the evolution of the shares of the major versions of these browsers relative to the total share of a given type of browser (all versions). Table 7.3 lists the release dates of major versions of these browsers.

7.2.2 Weekend Effect

For all browsers we observe a high frequency component on top of the much slower migration rate between major versions. This

Share of latest major version		
Browser	Share	Mio.
IE7	52.5%	579
FF2	92.2%	209
SF3	70.2%	34
OP9	90.1%	10
Total	**59.1%**	**832**

Table 7.2: *Share of the latest major version within a given type of browser as seen on Google's search and application Web sites in first week of June 2008.*

component is most pronounced in IE. We call this high frequency component the *weekend-effect*, as its periodicity follows exactly the weekly workday/weekend pattern throughout the year. The weekend effect can be explained by different preferences of end-users for major browser versions at work during the week and at home on the weekend. For example, IE7 consistently has a higher share at the weekends than during working days while the opposite is true for IE6. This relation is distorted over Christmas at the end of the year, supporting our interpretation. Since the end of December 2007 we observe a greater amplitude in the weekly pattern of IE6 and IE7. This could be explained with a sizable part of the IE population migrating over Christmas to new computers having Windows Vista and IE7 pre-installed. We find the weekend-effect for all major versions within the Firefox, Internet Explorer, Safari, and Opera population. Interestingly, we also found a *cross vendor weekend-effect* between Firefox and Internet Explorer, with Firefox being preferred over the weekend at the cost of the share of Internet Explorer.

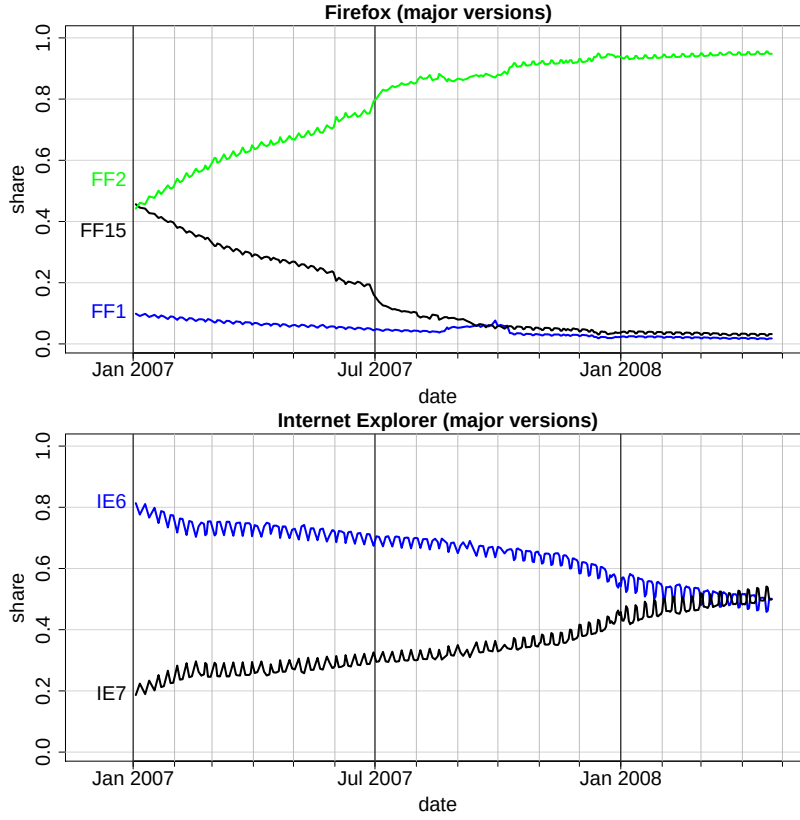

Figure 7.1: *Evolution of major version share for Firefox and Internet Explorer.*

7.2.3 Migration Drivers

On May 30th, 2007, FF1.5 reached the end-of-life (EOL) and Mozilla delivered the last security patch for FF1.5. Support should have ended in April but Mozilla extended the lifetime of FF1.5 in order to put in place a mechanism to allow FF1.5 users to upgrade to the latest FF2 release as soon as the upgrade

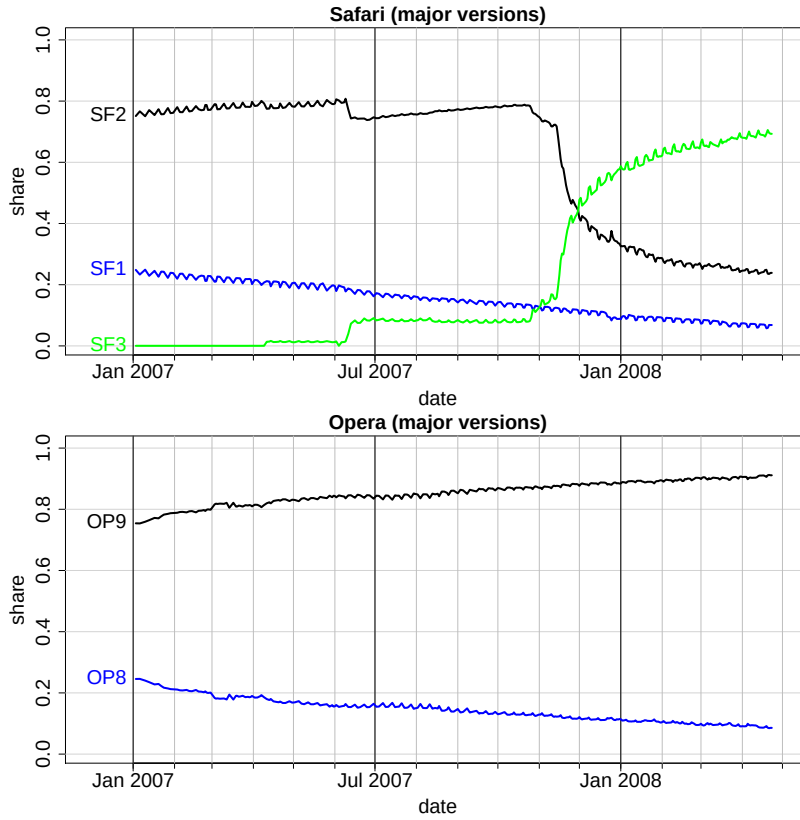

Figure 7.2: *Evolution of major version share for Safari and Opera.*

package is available. This delivery mechanism was included in version 1.5.0.12, the last update for FF15 on May 30th. Alternatively, users could manually upgrade to FF2 any time using Mozilla's download Web site. The impact of the EOL on the shares of FF2 and FF1.5 on May 30th is visible but smallish. Relatively few users chose to manually upgrade to FF2 as a result of the EOL of FF1.5. The release of the package to automatically

Browser	Version	Released
Internet Explorer	6.0	2001-08-27
Internet Explorer	7.0	2007-10-18
Firefox	1.0	2004-11-09
Firefox	1.5	2005-11-29
Firefox	2.0	2006-10-24
Safari	1.0	2003-06-23
Safari	2.0	2005-04-19
Safari	3.0	2007-10-26
Opera	8.0	2005-04-19
Opera	9.0	2006-06-20

Table 7.3: *Release of major browser versions.*

upgrade FF1.5 to FF2 on June 29th, 2007, has a much bigger impact as seen in Fig. 7.1. FF2 surpassed the combined share of FF1 and FF1.5 at the end of January 2007, about 15 month after its initial release (FF1.5 is considered as a major release).

Throughout our observation period, Microsoft supported both IE6 and IE7. We found no major discontinuities in their adoption rate. In March 2008 IE7 surpassed the share of IE6, 18 month after its initial release.

Apple released the first public beta of their Safari browser SF3 for Mac and Windows users on June 11th, 2007, which correlates with the first large increase in the SF3 share at the cost of SF2. Apple seems to have an enthusiastic beta tester community that readily adopted SF3 beta (gaining almost 10% in 3 days). Bundled together with Apple's new operating system Mac OS X Leopard (10.5), SF3[1] increases its market share on October 26th, 2007. However, the fastest rise in SF3 market share starts on November 16th, 2007, when SF3 is bundled with an auto-update of the then prevalent Mac OS X Tiger (10.4). SF3

[1] now in final version, out of beta

surpassed the share of SF2 in the last days of November 2007, which is very fast compared with 18 and 15 months for IE and FF respectively. However, in contrast to IE and FF, the share of the older SF1 was still above 10% at that time. This indicates that a large part of the Apple user population consistently misses major updates and still sticks to older versions of their browser. The Opera community is only slowly migrating to the next major version. More than 18 months after the release of OP9 we still found more than 10% share of OP8, which is no longer supported.

We find that bundling browser updates with existing automated update mechanisms has a major impact on migration speed. Upgrading to the next major version of a browser software is often intentionally delayed due to compatibility issues with critical i.e. intranet applications.

7.3 Minor Version Dynamics

7.3.1 Minor Version Dynamics

To analyze the "post-patch" risk period we plot the detailed update dynamics of minor versions of the free browsers Firefox and Opera released in 2007 in Fig. 7.3 and Fig. 7.4. Unlike Internet Explorer, these browsers provide minor version information in the user-agent string. The lack of minor version information therefore excludes Internet Explorer for this analysis. Further, we compare Firefox and Opera because both browsers are available for free, both are capable to run on multiple operating systems, and both are independent of any operating system vendor. Essentially, we measure the delay between the availability of a new update and the time it is installed by users. We find two distinct regimes of the adoption rate for both browsers: a very fast initial rise followed by a much slower continuing adoption thereafter. These two regimes are visualized in Fig. 7.5 where we plot the migration dynamics during the first 30 days after release, normalized to the

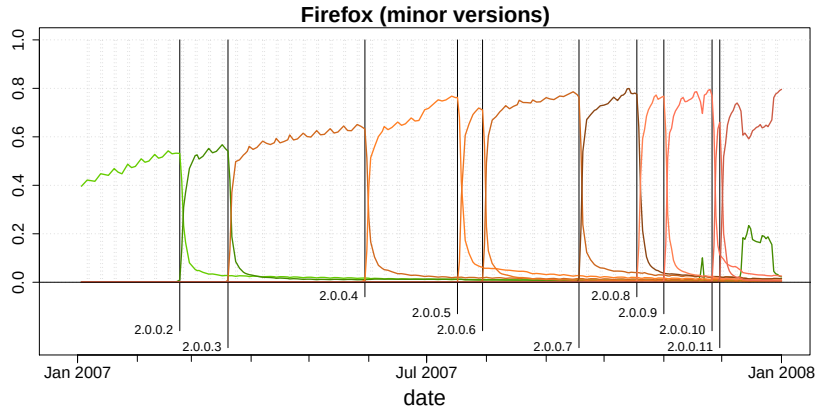

Figure 7.3: *Minor version update dynamics for Firefox in 2007, vertical lines depict the release of new minor versions.*

release date of respective version ($t = 0$). The sudden decline of a given minor version in Fig. 7.5 is attributed to versions that are superseded by the next minor version within the 30 days period plotted. After the first initial rise of the versions' share, the adoption rate is limited by the much slower migration of users *between major versions* of Firefox from FF 1.x to FF 2.x, as shown in Fig. 7.1 and Fig. 7.3. Note that 100% is the total of all versions of the respective browser. Some Firefox versions got replaced before 30 days, as seen in Fig. 7.3. Minor versions (N) and ($N-1$) (with (N) being the most recent version at any time) clearly dominate the dynamics of Firefox updates. The weekend is also visible as a high frequency oscillation in the update dynamics of minor versions. There is a striking difference with respect to the update dynamics within the Opera population. As with Firefox, we observe two distinct phases in the adoption rate. However, the initial adoption is much slower for Opera compared to Firefox. On average the first fast initial rise phase is about 11 days for Opera users and the share of

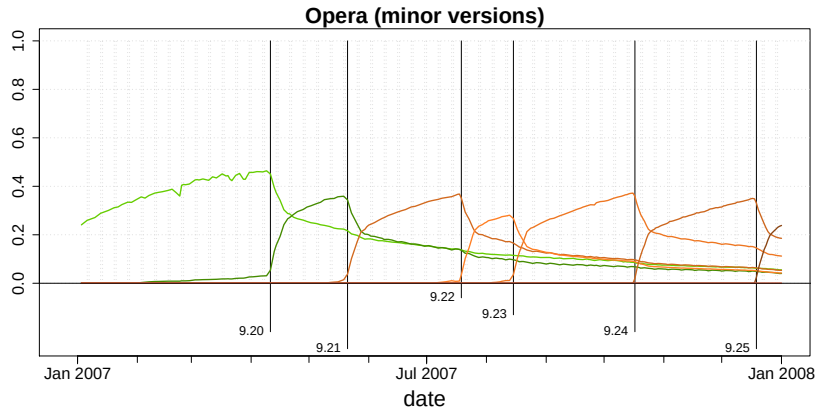

Figure 7.4: *Minor version update dynamics for Opera in 2007, vertical lines depict the release of new minor versions.*

the most recent version saturates at about 40%; well below the level achieved by the Firefox population. Further, older versions $(N-1, N-2, ..)$ persist remarkably long after the release of version (N), as illustrated in Fig. 7.6. After an initial fast decay, further loss of share of these older versions is very slow within the Opera population. This means that a considerable part of the Opera users stick to older, insecure versions of their browser. Opera reports that with all their different mobile platforms and the bandwidth restrictions that some of those platforms have, it didn't make sense to them to develop and deploy an auto-update mechanism [106]. Apparently getting security updates in the hands of its desktop users was just not priority enough for Opera. However, in December 2008 Opera announced that the next major version of their browser will finally include an auto-update mechanism.

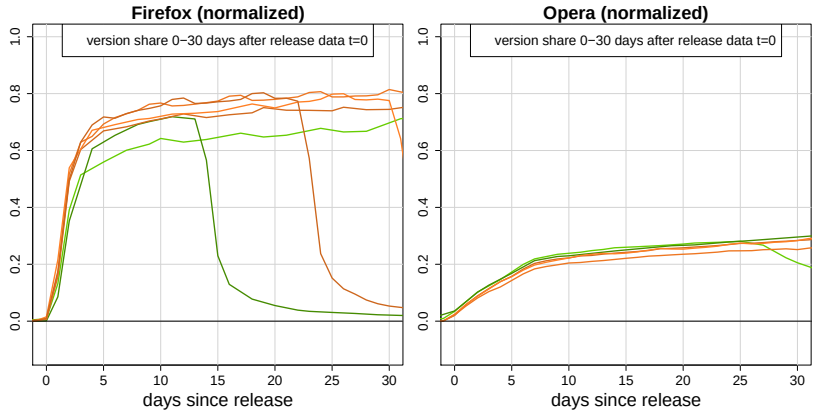

Figure 7.5: *Minor version share normalized to the release date* $t = 0$ *of the new version.*

"Post-patch" risk exposure

End users are exposed to security risks when surfing with an outdated version of their browser. To assess the extent and evolution of this risk exposure we measure the daily share of the latest (= most secure) browser version (N) for Firefox, Opera, and Safari between January 2007 to June 2008. Note that we are unable to measure the minor version of Internet Explorer. For the purpose of illustration we plot the share of the latest, most secure version of Firefox (whole year 2007) and Opera (close-up of the second half of 2007) in Fig. 7.6. These plots show the periods of increased risk of either browser in relation with the update cycle. We find that the *maximum share of the most secure web browser version* in active use never exceeded 47.6% for Internet Explorer, 83.3% for Firefox, 65.3% for Safari, and 56.1% for Opera between January 2007 to June 2008 as shown in Fig. 7.7. Note that we estimate the share of Internet Explorer, as described in the following section.

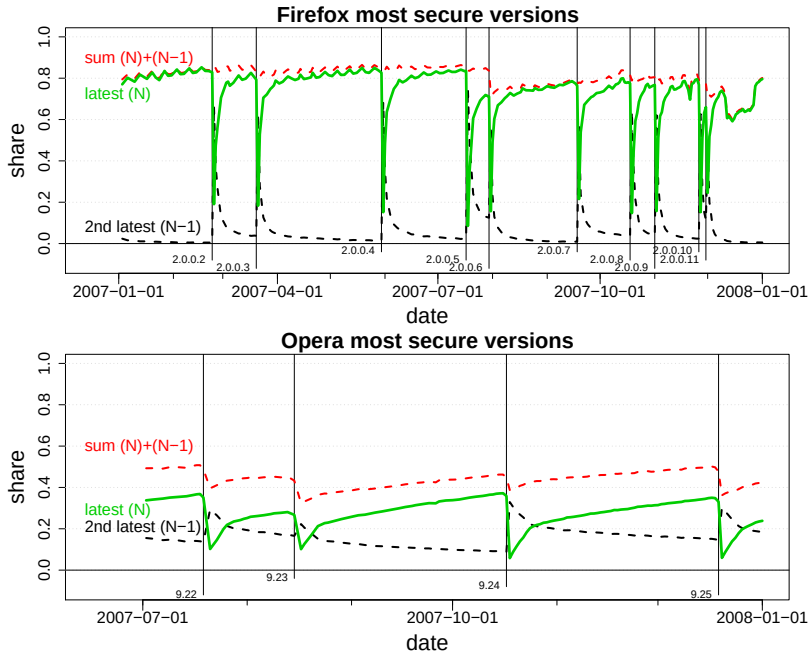

Figure 7.6: *Evolution of the share of the most recent versions* (N) *of Firefox and Opera. The dotted line depicts the previous version* $(N-1)$ *of the browser. 100% is the total of all versions of the respective browser share.*

Impact of Risk

In 2007, the Mozilla Foundation published 39 security advisories for Firefox and released ten new minor versions *2.0.0.2* to *2.0.0.11* of the browser FF2. Eight of these ten new versions address security vulnerabilities. Opera released six new versions in 2007, *9.20* to *9.25*, all of which fix security issues. Our measurement reveals that the adoption rate of minor versions is independent of the security risk being fixed, both for Firefox and Opera. E.g. there is no difference in the patch adoption rate

between high and low risk security updates. We conclude that the initial adoption rate is governed by the design and ergonomics of the auto-update functionality of the browser. Firefox can be updated with a single click if run with administrative rights. An update of Opera is essentially the same procedure as a complete manual download and install of the browser, typically requiring many user decisions and more than ten clicks.

7.4 Understanding the Web browser threat

Profit motivated cyber-criminals have rapidly adopted Web browser exploitation as a key vector for malware installation. With today's hostile Internet and drive-by download attack vectors[2], failure to apply patches promptly or missing them entirely exposes the host to infection and possibly subsequent data loss. In 2007, Google uncovered more than three million malicious Web addresses (URLs) that initiate drive-by downloads [107]. To capture the extent of this security problem we estimate the number of users worldwide not relying on the most recent Web browser version, which could result in a host compromise. We correlate the results of our measurements with data of Secunia's Personal Software Inspector (PSI) [108] to estimate the global population of Internet users not using the most recent version of their browser. Our measurement does not include the additional risk exposure of unpatched browser plug-ins or zero-day exploits.

Table 7.2 shows the usage share of the *latest major browser version* within each type of Web browser (e.g., the share of IE7 within the IE population). There were 1,408 million Internet users worldwide end of March 2008 [1]. Globally only 59.1%

[2]"drive-by" refers to the fact that Web browsers navigating to a benign (but compromised) site covertly download and executed malware.

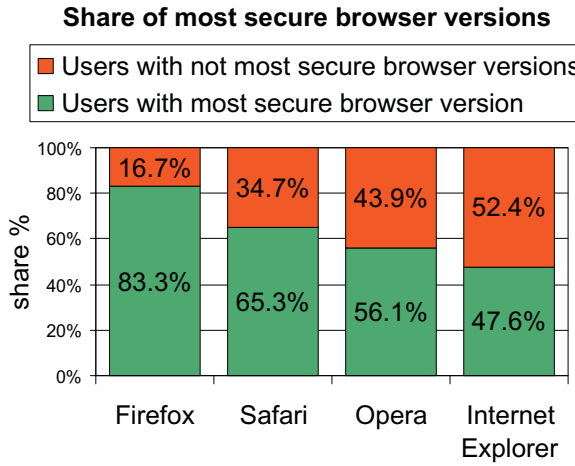

Figure 7.7: *Maximum share of users surfing the Web with the most secure versions of Firefox, Safari, Opera and Internet Explorer.*

(832 million users), make use of the latest major version of their preferred Web browser to navigate the Internet. This is an estimate for the *upper bound* for the global share of the most secure browsers in use. However, 576 million users surfed the Internet without using the latest major version of their preferred browser.

Most secure browser version

In this dissertation, the term **most secure browser** designates the latest official public release of a vendor's Web browser at a given date. Beta versions are not considered an official public release. We use the most recent major versions of Internet Explorer 7, Firefox 2, Safari 3, and Opera 9 as the benchmark version for our most secure Web browser measurements. Microsoft's

Internet Explorer version 6, independent of its patch level, is not considered the most secure version of Internet Explorer by Windows expert Brian Livingston [109] and even Microsoft calls IE7 "*an extremely important update from a security perspective*" over IE6 and states "*There are dangers that simply didn't exist back in 2001, when Internet Explorer 6 was released to the world. Internet Explorer 7 makes surfing the web fundamentally safer by offering greater protection against viruses, spyware, and other online risks.*" [110].

Estimating number of users at risk

Analysis of the distribution of patch implementations within the *latest major version* as of Table 7.2 is used to measure the share of the most secure version for each browser type. For Firefox, Safari, and Opera we use the HTTP user-agent information in the Google's Web log data sets to determine the minor version. For Internet Explorer we rely upon the results of Secunia's PSI statistics [108] to estimate the share of the most secure version. Secunia PSI identified (for the month of May 2008) that 4.4% of IE7, 8.1% of Firefox, 14.3% of Safari (Windows only), and 15.2% of Opera users have not applied the most recent security patches available to them from the software vendor [108]. In comparison, we observe that 16.7% of Firefox, 34.7% of Safari (all OS), and 43.9% of Opera Web browser installations (using our Web server log-based measurements) have not applied the most recent security patches. We find that our Firefox, Safari, and Opera results were higher than those of Secunia's, differing by a factor of 2.1 (Firefox), 2.4 (Safari), and 2.9 (Opera), and attribute this difference to a probable bias for more security aware users to take advantage of Secunia's security scanner PSI than the average global community. To derive the global population of users with browsers vulnerable to built-in vulnerabilities we use the results of our measurements for Firefox, Safari, and

	Latest Major		Estimate A		Estimate B	
Browser	%	Mio.	%	Mio.	%	Mio.
IE	41.1%	578.7	4.4%	25	9.2%	53
FF	16.1%	226.7	16.7%	38	16.7%	38
SF	3.4%	47.9	34.7%	17	34.7%	17
OP	0.8%	11.3	43.9%	5	43.9%	5
Total	98.6%	1,388.3	43.3%	609	45.2%	637

Table 7.4: *Estimation of the number of users not using the most secure version of their browser.*

Opera. We estimate the value for Internet Explorer based upon the findings of Secunia as shown in Table 7.4. We present two estimates: (A) for which we take the IE7 share from Secunia (4.4%) and (B) where we correct the IE7 share from Secunia using the lowest factor previously found ($2.1 \cdot 4.4\%$). Note that "Latest Major" in 7.4 depicts the share of the latest major version of a given browser in daily use as of Table 7.1.

- *Estimate A*
  Firefox, Safari, and Opera shares are from our Google Web log measurements. The IE7 share of 4.4% is from Secunia's PSI measurement [108]. This is a minimum estimate as Secunia's PSI measurement is likely biased towards more security aware users. IE6 is not considered a most secure Web browser version [110].

- *Estimate B*
  We apply the factor 2.1 to the IE7 share (2.1 x 4.4% = 9.2%) to correct for the bias of Secunia's PSI measurement within a security aware user population. The factor was found when comparing Firefox, Safari, and Opera data from Google log files with Secunia's data.

Our Estimate B shows that at least 45.2%, or 637 million users, were not using the most secure Web browser version on any day from January 2007 to June 2008. These browsers are an easy target for drive-by download attacks as they are potentially vulnerable to known exploits.

We believe that our measurement of potentially insecure Web browsers based upon major and minor version information is smaller than the global number of users at risk. Insecure Web browsers (i.e., they have "built-in" vulnerabilities and security weaknesses) are of course a critical security problem, but vulnerable plug-ins that are accessible (and exploitable) through the Web browser extend the risk exposure. Our measurements are limited to the information available in the user-agent string, hence we can't directly measure the number of users having out of date and vulnerable Web browser plug-ins. However, there is public evidence that this number adds to the number of users with browsers having "built-in" vulnerabilities: A typical Web browser has more than one plug-in application installed. Media players and other plug-ins are ubiquitous, with individual usage shares frequently exceeding 80% [111]. Table 7.5 lists the adopted use of some of the most popular plug-in applications - all of which are accessible through a Web browser.

Plug-In	**Vendor**	**Share**	**Support**
Flash Player	Adobe	98.8%	all
Java	Sun	84.0%	all
Media Player	Microsoft	82.2%	IE only
QuickTime Player	Apple	66.8%	all
Shockwave Player	Adobe	55.6%	all
RealOne Player	Real Networks	47.1%	all
Acrobat PDF Reader	Adobe	>80%	all

Table 7.5: *Usage shares of some widely used plug-ins.*

Considering our analysis of insecure Web browser usage, we deem it unlikely that the same users achieve higher patch levels for multiple plug-ins installed; with each plug-in relying on different patching and updating mechanisms. For example, Secunia's PSI states 18.7% of all WinAMP 5 installations miss important security updates, and 21.7% of all Quicktime 7 installations are out of date.

Therefore, we believe that globally more than 45.2%, or 637 million users, found in Estimate (B) are potentially vulnerable to attack.

7.5 Summary

Although Web browser users may wish for perfect software that will never have any exploitable software vulnerabilities, the nearest they can realistically hope for is that any vulnerabilities are promptly fixed by the software vendors and instantly applied to their browser. The "post-disclosure" and "post-patch" risk exposures introduced in this thesis measure these properties. Critical to this instantaneous patching process is the mechanism of "auto-update". Our measurement confirmed that Web browsers which implement an internal auto-update patching mechanism do much better in terms of faster update adoption rates than those without. Our comparison of the update dynamics between Firefox and Opera identified that auto-update mechanisms are crucial for timely patching. We showed that automatic mechanisms to deliver upgrades (e.g., bundling a new major version with an operating system upgrade) outperform more complex and less timely solutions requiring user interaction and that the severity of a vulnerability has no influence on the patch adoption speed. Firefox's auto-update was found to be way more effective than Opera's manual update download reminder strategy.

In our measurement period from January 2007 to June 2008, most users updated to a new version of Firefox within three days of a new public release, resulting in up to 83% of users having the most current and secure Firefox version installed. It took users of the Opera Web browser an average of 11 days before reaching an update saturation at a level of up to 56%. While Firefox and Opera check for updates when the browser is used, Safari relies on an external Apple-updater that appears to only poll for new updates at scheduled regular intervals while Internet Explorer gets updated as part of the monthly distributed Windows patches.

For years the software industry has promoted one security best practice over all others: *always use the most recent version of the installed software and instantly apply the latest patches.* However, as our measurements demonstrate, without proper automation this approach alone is doomed to fail. Further, in light of an increasing number of plug-ins and software from different vendors operating on the same host, this approach does not scale without better integration between vendors. If many different auto-update implementations on the same host keep software and plug-ins from n different vendors up to date, then the user gets confused at best. Further, the added complexity of having n different update mechanisms does not help either to make the host more robust or less vulnerable.

Chapter 8

Conclusions and Discussion

8.1 Summary of Contributions

To understand the processes and limitations behind the technology driven evolution of our economy and society, knowledge on how security information is handled at large becomes of interest. In this dissertation we claim that knowledge of the vulnerability lifecycle (the vulnerability discovery-, exploit-, disclosure-, and patch-time) allows us to distinguish major processes in the security environment and to quantify the risk exposure and evolution thereof at macroscopic level. Our model of the *security ecosystem* in Chapter 3 demonstrates how events of the vulnerability lifecycle can be related to major processes in the security ecosystem. Measurements of what we call the dynamics of (in)security provided insight into the prevalence of processes such as *responsible disclosure*, the *patch release performance* of vendors, or identified the drivers behind effective auto-update strategies. How security information is handled at large has an important impact on the economic incentives present in the

security ecosystem. Therefore, we analyzed the major sources of security information, measured their performance in Chapter 4, and made out their important role for the functioning of the security ecosystem.

This thesis made the following contributions:

8.1.1 Model of the Security Ecosystem

We introduced a model to represent the processes and the flow of security vulnerability information between the main players in the security ecosystem on a macroeconomic scale. We then provided a formal introduction and definition of the vulnerability lifecycle and associated events in the lifecycle to processes in the security ecosystem model. To permit effective measurements we introduced the concept of vulnerability lifecycle normalization with respect to the disclosure date; and provided a concise definition of what is considered the disclosure date of a vulnerability.

8.1.2 Security Information Provider

We provided the first analysis of the primary security information sources the industry relies on. We analyzed several *Security Information Providers (SIP)* and demonstrated that independent and trusted SIPs act like the free press in an open society; they are efficient watchdogs to expose important issues to the public.

8.1.3 Empirical Evaluation

Covering more than 27,000 vulnerabilities published from 1996 to 2007 we carried out the first large scale analysis of the vulnerability lifecycle. Thereby we measured the state and the evolution of key processes of the security ecosystem and identified trends. We presented the first empirical study documenting the

prevalence of the "responsible disclosure" process and developed the *Zero-day patch* and *overdue patch* metrics to capture and compare different vendors' patch release performance at large. We introduced a new method to passively measure Web browser patch dynamics at a global scale and provided empirical data covering approximately 75% of the world's Internet users for more than a year. The presented methods of measurements rely on publicly available data only. While Google's Web server logs used for the evaluation in Chapter 7 are not publicly available, the methodology introduced for these measurements can be applied by anyone operating a Web server with reasonably large traffic volumes. Our measurements can be carried out and reviewed by anyone interested in the topic.

8.2 Critical Assessment

Despite processing more than half a million Web pages and tapping into several security databases, our measurements have some limitations:

- *Controlled experiment.* The security landscape is a fast and ever-changing place. It is therefore not possible to execute a controlled experiment at large scale. The methods presented in this thesis rely on few assumptions thought to be reasonable and robust at the time of writing. E.g. we rely on the CVE database succeeding to catalog a sizable part of vulnerabilities of relevance, on SIPs providing accurate and independent information to the public, and on the NVD to provide an accurate mapping of CVE to vendor name. Should any of these assumptions become obsolete in the future, a reassessment of the measurement methods presented in this thesis will be necessary.

- *Availability of information.* Cyber-criminals and members of the security underground scene do not provide or disclose information of their operations. At best we can estimate or deduce the (minimum or maximum) extent of certain characteristics based on the measurement of visible effects of their operation.

- There is more data openly available than what we used for our measurements. For an analysis of the present scale we could only account for data that was accessible with some minimal degree of automation. For example, correlating exploit material to CVE identifiers through manual code analysis was no option.

- *Limited real time processing.* Some of our measurements do not provide accurate real time results as they rely on information available in the future only (e.g., the discovery date of a vulnerability is only made public upon disclosure).

- *Risk exposure time.* Our measurements estimate the risk exposure time at large scale and not the security risk of a given group. Analyzing the true risk always depends on an individual assessment of different risk factors and expected losses. An evaluation of the true risk at large scale is infeasible due to the inaccessibility, privacy, or unavailability of individual data.

8.3 Concluding Remarks and Future Work

Access to Google's global Web server logs enabled us to provide the first in-depth, global perspective on the state of insecurity for Web browser technologies. Understanding the nature of the threats against Web browsers and their plug-in technologies is

important for continued safe Internet usage. By measuring the patching processes of Web browser user populations, we have been able to identify the potential malicious exploitation of Web browser technologies on global scale and proved how existing mechanisms such as Firefox's auto-update can outperform more complex and less timely solutions. We quantified the lower bounds of the Web browser population vulnerable to attacks through security weaknesses. We found 637 million (or 45.2%) Internet users at risk worldwide due to not running the latest most secure browser version [10]. We believe that, in the majority of cases, the absence of critical or important updates to the Web browsers can be attributed to three important factors; technological (*can't do*), motivational (*don't care*), or informational (*don't know*).

We believe that new strategies could be developed in the near future to increase both host protection and user awareness:

- Introduction of a "Best Before" date concept
- Standardization of "Auto-Update" mechanisms

8.3.1 "Best Before" Date Concept

A critical path to increasing the security of Web browsers (indeed any and all inter-networked applications including online game clients) involves making the user aware of the risk they are exposing themselves and their host to, but without introducing additional complexity:

Almost all users are familiar with the concept of "sell by", "expires on", or "best before" date stamps on perishable goods. Consumers tend to rely on this date information in order to decide whether to purchase the goods, when to use the goods and when to dispose of the goods. Once a particular perishable good has exceeded its "best before" date, the consumers are forced to evaluate their personal risk to using it or disposing of it. The

greater the lapse between the "best before" date and the current date, the more risk the consumer assumes by not disposing of it. Given the state of the software industry and the growing threat of exploitable vulnerabilities within all applications (not just Web browsers), we believe that the establishment of a "best before" date for all new software releases could prove an invaluable means to educating the user to patch or "refresh" their software applications. The same "best before" date information could also be leveraged by Internet services to help evaluate or mitigate the risk of customers who are using out of date software and are consequently at a higher risk of having been compromised.

A public mindset change is required to counter evolving Internet threats, and a "best before" dating system would make visible the risks of using out-dated and insecure software.

In order to achieve a viable "best before" dating system, software vendors need to follow stricter practices in the allocation of version number information and make those version numbers more accessible.

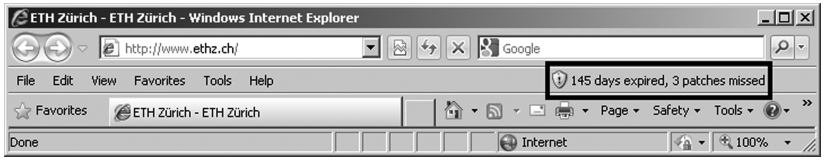

Figure 8.1: *Illustration of "best before" implementation on Web browser.*

For example, an online banking service may use the version information supplied by the user's Web browser to establish when the software was last updated and to assess the level of risk the host has been compromised with malware. Armed with that information, the banking application may decide to implement

additional safeguards and inspection on subsequent transactions by the user.

We believe that the "best before" dating concept could be built into most existing software applications, and thereby provide a convenient and persistent validation of the likely integrity of the software. For example, popular Web browsers could display a visual warning of expiry and how many patches are currently missing as illustrated in Fig. 8.1.

8.3.2 Standardization of Auto-Update Mechanisms

The constant discovery of new vulnerabilities for almost every kind of software is a reality we face today. In this hostile environment, and backed by our findings, we come to conclude that software generally should to be considered as a "perishable good". From this perspective follows that "auto-update" functionality should be integrated in all types of software by design, not as a later add-on or option. In light of the increasing number of plug-ins and software from different vendors operating on the same host, todays "one update mechanism per vendor" approach does not scale. At best it confuses the user and adds unnecessary complexity to the system. Further, embedded systems (ADSL/cable modems, printers, network cams, PCS[1], toasters, wearable computers) become ubiquitous and get networking capability at a faster pace than they get a reasonable update mechanisms. However, it is inefficient and in general economically prohibitive for the different engineering teams of software applications, embedded systems, and plug-ins to each develop independent solutions for the same problem. Further, there is the risk that such an approach would result in many insecure implementations that potentially do more harm than good. We believe that standard mechanisms and protocols should be developed to make secure,

[1] Process Control System

scalable, and affordable auto-update functionality easily available for developers of various types of software. This could include the setup of vendor independent trusted services to ensure the timely and authenticated roll out of updates, e.g. provided by CERTs.

8.3.3 Continued Measurement

Our society is still in an early phase of the adoption of the new and seemingly endless opportunities of information technology. During the early embryonic phase of innovation, before the emergence of a dominant design, the industry is characterized by high levels of experimentation among producers and customers. "*The market and the industry are in a fluid stage of development. Everyone - producers and customers - is learning as they move along.*" [13]. The security industry is not yet formally described and evaluated at large scale. Up to now measurements have typically focused on partial analysis of individual events. In this dissertation we presented methods to model and measure important processes in the security ecosystem at macroeconomic scale. Our datasets merely cover the first few years of the embryonic phase of information security. Such measurement not only help understanding the state of information security; applied to future data our metrics can provide insight into the success and limitation of processes (e.g., "where does vendor patch performance saturate?", "what is the role of vulnerability markets") and identify trends (e.g., "can improved software technology delay exploit development?"). Technological measures have an important role to play in a wide variety of economic investigations and in attendant efforts towards policy formulation. "*In a nutshell, we need unequivocal ways of measuring technology so as create public awareness of innovations and to ensure consumer sovereignty*" [112]

Appendix A

A.1 Security Information Sources

A.1.1 Security Information Providers (SIP)

US-CERT (CERT)

Worldwide, there are more than 250 organizations that use the name *CERT* or a similar name that deal with cyber security. The first of these types of organizations is the CERT Coordination Center CERT/CC, established by DARPA at Carnegie Mellon University in December 1998 [113], to address computer security concerns of research users of the Internet. The US Computer Emergency Readiness Team US-CERT [114] is the operational arm at the United States Department of Homeland Security (DHS). US-CERT publishes information about a wide variety of vulnerabilities as *Vulnerability Notes*. Vulnerability notes include technical descriptions of the vulnerability, as well as the impact, solutions and workarounds, and lists of affected vendors. Vulnerabilities that meet a certain severity threshold are described in *Technical Alerts*. A number between 0 and 180 assigns an approximate severity to the vulnerability. We use the vulnerability notes of US-CERT for our research. US-CERT is represented in the CVE Editorial Board.

US-CERT Web site: `http://www.kb.cert.org/vuls/`

SecurityFocus (SF)

SecurityFocus is a security news portal and purveyor of information security services since 1996. SecurityFocus is the owner of the well-known Bugtraq [115] mailing list as of 1999. In August 2002, Symantec [116] acquired SecurityFocus in full. Part of the purchase agreement was to keep SecurityFocus as an independent security portal. Symantec offers managed security services (MSS) and builds a range of security products, for end-users and enterprises (e.g., anti-virus, intrusion prevention systems). Security advisories and exploit material are provided to the public through the SecurityFocus vulnerability database, which is not equal to BugTraq (the mailing list) or alerts of the Symantec security response team. SecurityFocus assigns no risk rating but classifies the type of vulnerability. Symantec and SecurityFocus are represented in the CVE Editorial Board.

SecurityFocus Web site: `http://www.securityfocus.com`

IBM Internet Security Systems - X-Force (XF)

The X-Force is the security research and development group of Internet Security Systems (ISS), since 2006 part of IBM. IBM ISS offers a range of security products and services, namely Managed Security Services (MSS), Intrusion Prevention Systems (IPS) and enterprise vulnerability scanner. IBM X-Force does active research of diverse products and technologies and ongoing surveillance within the security scene to identify new trends and malware. Since 1996 the X-Force publishes relevant discoveries as security advisories [77, 117] in their X-Force Database (XFDB). X-Force assigns one of three possible risk levels to vulnerabilities: High, Medium, Low. IBM and Internet Security Systems are represented in the CVE Editorial Board.

X-Force Web site: `http://xforce.iss.net`

Secunia (Secunia)

Secunia [118] was founded in 2002 and is based in Denmark It is an independent provider of vulnerability intelligence in the sense that it does not sell protection products. Aside from gathering information from external sources, Secunia also conducts its own internal research. Secunia hosts the *full disclosure* security mailing list [119], which is an unmoderated high-traffic forum for the disclosure of security information. The list was founded 2002 (after Symatec bought SecurityFocus) as an alternative to the moderated Bugtraq mailing list. Secunia assigns a five level risk rating to vulnerabilities: Not Critical, Less Critical, Moderately Critical, Highly Critical, and Extremely Critical. Secunia is not represented in the CVE Editorial Board.

Secunia Web site: `http://secunia.com`

French Security Incident Response Team (FrSIRT)

The French Security Incident Response Team FrSirt [120] is a private company based in southern France founded in 2003. FrSirt started delivering security and exploit advisories to the public in 2005. However, since early 2006 exploit information is only available as a paid service. FrSIRT provides a four level risk rating of the considered vulnerabilities. FrSIRT is not represented in the CVE Editorial Board and has renamed itself to become VUPEN Security (VUlnerability management and PENtesting) in December 2008. VUPEN is aiming at future international expansion, and says an investment fund has taken a stake in the company as a source of funds. The company feels its new name reflects its business better than the previous one. This is not the company's first change of name: it previously offered exploits and security advisories under the name of K-otic. Shortly

after rebadging as FrSIRT, the company provided its exploits only as part of a paid Vulnerability Notification Service (VNS).

FrSIRT Web site: `http://www.frsirt.com`

SecurityTracker (SecTrack)

SecurityTracker [121], started in 2001, is dedicated solely to reporting on security vulnerabilities and does no not conduct its own security research. SecurityTracker is a vendor neutral security portal, deploying automated agents to scan web sites, e-mail lists, newsgroup feeds, vendor bulletins, and incident advisory sources for the latest vulnerability information. Security advisories published by SecurityTracker are not risk rated, they classify the vulnerability impact with 13 classes. SecurityTracker is not represented in the CVE Editorial Board.

SecurityTracker Web site: `http://www.securitytracker.com`

SecurityWatch (SecWatch)

SecWatch [122] provides the security community with vulnerability and exploit information since 2004. SecWatch was considering the sale of its site and related services as of April 2008 and stopped serving security advisories by end of May 2008. SecWatch provides a five level risk rating with security advisories. SecWatch is not represented in the CVE Editorial Board.

SecWatch Web site: `http://secwatch.org`

A.1.2 Other Security Information Sources

Open Source Vulnerability Database (OSVDB)

OSVDB is an independent and open source database created by and for the community. The goal of the project is to provide

accurate, detailed, current, and unbiased technical information on security vulnerabilities. The project promotes open collaboration between companies and individuals to reduce expenses inherent with the development and maintenance of in-house vulnerability databases. The project was started in August 2002 at the Blackhat [123] and DEFCON [124] conferences by several industry notables. Under mostly-new management, the database officially launched to the public on March 31, 2004. The Open Security Foundation (OSF) [125] was created to ensure the project's continuing support. Vulnerability reports, advisories and exploits posted in various security lists enter the database as a new entry. After new entries are analyzed and refined, descriptions of the vulnerability, its solutions and test notes are added, reviewed and published.

OSVDB Web site: `http://www.osvdb.org`

BugTraq

BugTraq is a full disclosure moderated mailing list for the detailed discussion and announcement of computer security vulnerabilities: what they are, how to exploit them, and how to fix them. The Bugtraq mailing list was created in 1993 in response to the perceived failings of the existing Internet security infrastructure of the time. It started as a unmoderated mailing list for the *full disclosure* of security vulnerabilities, to become moderated in 1995. Bugtraq was originally hosted at Crimelab.com. It was moved to the Brown University NetSpace Project – which has since been reorganized as the NetSpace Foundation – on June 5, 1995, the same day that its moderation began. In July 1999 it became the property of SecurityFocus and was moved there. SecurityFocus was acquired in full by Symantec on August 6, 2002.

BugTraq Web Site: `http://www.securityfocus.com/archive/1`

A.1.3 Exploit Information Sources

Metasploit Project

The Metasploit Project is a computer security project which provides information about security vulnerabilities and aids in penetration testing and IDS signature development. Metasploit was created in 2003. It is known for releasing technically sophisticated exploits to public security vulnerabilities. In addition it is a powerful tool for third party security researchers to investigate potential vulnerabilities. Like many information security tools, Metasploit can be used for both legitimate and unauthorized activities. Metasploit's position as vulnerability development framework has frequently led to the release vulnerability advisories accompanied by a third party Metasploit exploit modules that highlights the exploitability, risk, and remediation steps of that particular bug.

Metasploit Web Site: `http://www.metasploit.com`

Bibliography

[1] Internet World Statistics, "World Internet Users and Population Statsitics," 2008, http://www.internetworldstats.com/stats.htm.

[2] R. Anderson, "Information Security Economics - and Beyond," in *DEON '08: Proceedings of the 9th international conference on Deontic Logic in Computer Science*, 2008, pp. 49–49.

[3] ——, "Why information security is hard - an economic perspective," in *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, 2001, pp. 358–365.

[4] A. Shostack and A. Stewart, *The new school of information security.* Addison-Wesley Professional, 2008, ISBN:9780321562753.

[5] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in *LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, 2006, pp. 131–138.

[6] S. Frei and M. May, "The speed of (in)security," BlackHat 2006 USA, 2006, http://www.blackhat.com/presentations/

bh-usa-06/BH-US-06-Frei-May.pdf.

[7] Stefan Frei, Dominik Schatzmann, Bernhard Plattner, Brian Trammel, "Modelling the Security Ecosystem - The Dynamics of (In)Security," in *Workshop on the Economics of Information Security (WEIS)*, 2009.

[8] Stefan Frei and Martin May, "Putting private and government CERT's to the test," in *20th Annual FIRST Conference, June 22-27, 2008, Vancouver, Canada*, 2008.

[9] S. Frei, B. Tellenbach, and B. Plattner, "0-Day Patch - Exposing Vendors (In)security Performance," 2008, http://www.blackhat.com/presentations/bh-europe-08/Frei/Whitepaper/bh-eu-08-frei-WP.pdf.

[10] S. Frei, T. Dubendorfer, G. Ollmann, and M. May, "Understanding the Web browser threat," ETH Zurich, Tech. Rep. 288, 2008, http://www.techzoom.net/insecurity-iceberg.

[11] T. Duebendorfer and S. Frei, "Why Silent Updates Boost Security," in *CRITIS'09: 4th Int. Workshop on Critical Information Infrastructures Security*, 2009.

[12] S. Frei, T. Duebendorfer, and B. Plattner, "Firefox (In)Security Update Dynamics Exposed," *Computer Communication Review*, vol. 39, no. 1, 2009.

[13] J. M. Utterback, *Mastering the dynamics of innovation.* Boston, MA, USA: Harvard Business School Press, 1996.

[14] M. W. Oborne, *The Security Economy.* OECD, Paris, 2004, ISBN 92-64-10772-X.

[15] R. Anderson and T. Moore, "The Economics of Information Security," *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

[16] T. A. Longstaff, C. Chittister, R. Pethia, and Y. Y. Haimes, "Are We Forgetting the Risks of Information Technology?" *Computer*, vol. 33, no. 12, pp. 43–51, 2000.

[17] L. A. Gordon and M. P. Loeb, "Using information security as a response to competitor analysis systems," *Commun. ACM*, vol. 44, no. 9, pp. 70–75, 2001.

[18] S. L. Pfleeger, R. Rue, J. Horwitz, and A. Balakrishnan, "Investing in Cyber Security: The Path to Good Practice," *The RAND Journal*, vol. Vol 19, No. 1, 2006.

[19] N. G. Carr, "IT Doesn't Matter," *Harvard Business Review*, pp. 41–49, 2003.

[20] T. Dubendorfer, A. Wagner, and B. Plattner, "An Economic Damage Model for Large-Scale Internet Attacks," in *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04)*, 2004, pp. 223–228.

[21] US-CERT, "Vulnerability Statistics," http://www.cert.org/stats/vulnerability_remediation.html.

[22] NIST, "National Vulnerability Database - Vulnerability Statistics," http://web.nvd.nist.gov/view/vuln/statistics.

[23] M. Greenwald, C. A. Gunter, B. Knutsson, A. Scedrov, J. M. Smith, and S. Zdancewic, "Computer Security is Not a Science (but it should be)," in *Large-Scale Network Security Workshop*, 2003.

[24] MITRE, "CVE Vulnerability Terminology," http://cve.mitre.org/about/terminology.html.

[25] C. V. Lundestad and A. Hommels, "Software vulnerability due to practical drift," *Ethics and Inf. Technol.*, vol. 9, no. 2, pp. 89–100, 2007.

[26] Y. K. Malaiya and J. Denton, "Module Size Distribution and Defect Density," in *ISSRE '00: Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE'00)*, 2000, p. 62.

[27] P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz, "An Empirical Study of Software Reuse vs. Defect-Density and Stability," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, 2004, pp. 282–292.

[28] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 3, pp. 309–346, 2002.

[29] W. A. Arbaugh, W. L. Fithen, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," *Computer*, vol. 33, no. 12, pp. 52–59, 2000.

[30] A. Arora, R. Krishnan, A. Nandkumar, R. Telang, and Y. Yang, "Impact of Vulnerability Disclosure and Patch Availability – An Empirical Analysis," in *Workshop on the Economics of Information Security (WEIS 2004)*, 2004.

[31] H. C. Hasan Cavusoglu and S. Raghunathan, "Emerging issues in responsible vulnerability disclosure," in *WITS*, 2004.

[32] K. Kannan and R. Telang, "An economic analysis of market for software vulnerabilities," in *Workshop on the Economics of Information Security (WEIS 2004)*, 2004.

[33] Qualys, "Laws of Vulnerabilities," 2005, http://www.qualys.com/docs/Laws-Report.pdf.

[34] OISA Organization for Internet Safety, "Guidelines for Security Vulnerability Reporting and Response," http://www.oisafety.org/guidelines/.

[35] A. Arora, R. K. R. Telang, and Y. Yang, "An Empirical Analysis of Vendor Response to Disclosure Policy," in *Workshop on the Economics of Information Security (WEIS 2005)*, 2005.

[36] R. T. Ashish Arora, Ramayya Krishnan and Y. Yang, "Empirical analysis of software vendors patching behavior, impact of vulnerability disclosure," Carnegie Mellon University, Tech. Rep., 2006.

[37] FIRST, "Forum of Incident Response and Security Teams," http://www.first.org.

[38] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, pp. 5–83, 1883.

[39] R. Anderson, "Security in open versus closed systems - the dance of boltzmann, coase and moore," in *In Conference on Open Source Software Economics*, 2002, pp. 1–15.

[40] B. Schneier, "The nonsecurity of secrecy," *Commun. ACM*, vol. 47, no. 10, p. 120, 2004.

[41] J. Bollinger, "Economies of Disclosure," *SIGCAS Comput. Soc.*, vol. 34, no. 3, pp. 1–1, 2004.

[42] B. Schneier, "Locks and Full Disclosure," *IEEE Security and Privacy*, vol. 01, no. 2, p. 88, 2003.

[43] A. Hobbs, *Locks and Safes: The Construction of Locks*, C. Tomlinson, Ed. Virtue & Co., London, 1853,1868.

[44] A. Ozment and S. E. Schechter, "Milk or wine: does software security improve with age?" in *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, 2006.

[45] E. Rescorla, "Is Finding Security Holes a Good Idea?" *IEEE Security and Privacy*, vol. 3, no. 1, pp. 14–19, 2005.

[46] A. Arora, R. Telang, and H. Xu, "Optimal Policy for Software Vulnerability Disclosure," in *Workshop on the Economics of Information Security (WEIS 2004)*, 2004.

[47] S. Ragan, "MiFare Classic vulnerability fully and officially exposed," 2008, http://www.thetechherald.com/article.php/200841/2208/MiFare-Classic-vulnerability-fully-and-officially-exposed.

[48] Electronic Frontier Foundation EFF, "Coders' Rights Project Vulnerability Reporting FAQ," http://www.eff.org/issues/coders/vulnerability-reporting-faq.

[49] Computer Crime Research Center, "Cybercrime definition," Computer Crime Research Center, 2006, http://www.crime-research.org/articles/joseph06.

[50] P. Williams, "Organized Crime and Cybercrime: Synergies, Trends, and Responses," Computer Crime Research Center, http://www.crime-research.org/library/Cybercrime.htm.

[51] McAfee, "Virtual Criminology Report 2008," 2008, http://www.mcafee.com/us/research/criminology_report/default.html.

[52] B. Thomas, J. Clergue, A. Schaad, and M. Dacier, "A comparison of conventional and online fraud," in *CRIS'04,*

2nd International Conference on Critical Infrastructures, October 25-27, 2004 - Grenoble, France, 2004.

[53] Gunter Ollmann, "Continuing business with malware infected customers," 2008, http://www.technicalinfo.net/papers/MalwareInfectedCustomers.html.

[54] E. Levy, "Approaching Zero," *IEEE Security and Privacy*, vol. 2, no. 4, pp. 65–66, 2004.

[55] J. Radianti and J. J. Gonzalez, "Understanding Hidden Information Security Threats: The Vulnerability Black Market," *Hawaii International Conference on System Sciences*, vol. 0, p. 156c, 2007.

[56] PandaLabs, "MPack uncovered, Lab Report," http://blogs.pandasoftware.com/blogs/images/PandaLabs/2007/05/11/MPack.pdf.

[57] M. Whipp, "Black market thrives on vulnerability trading," PCpro, 2006, http://www.pcpro.co.uk/news/84523/black-market-thrives-on-vulnerability-trading.html.

[58] iDefense, "Vulnerability Contributor Program," http://labs.idefense.com/vcp.

[59] TippingPoint, "Zero Day Initiative," http://www.zerodayinitiative.com/.

[60] D. McKinney, "Vulnerability Bazaar," *IEEE Security and Privacy*, vol. 5, no. 6, pp. 69–73, 2007.

[61] F. Swiderski and W. Snyder, *Threat Modeling*. Redmond, WA, USA: Microsoft Press, 2004.

[62] O. H. Alhazmi, Y. K. Malaiya, and I. Ray, "Security vulnerabilities in software systems: A quantitative

perspective," in *Proc. Ann. IFIP WG11.3 Working Conference on Data and Information Security*, 2005, pp. 281–294.

[63] A. Ozment, "Improving vulnerability discovery models," in *QoP '07: Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[64] F. F. Lindner, "Software security is software reliability," *Commun. ACM*, vol. 49, no. 6, pp. 57–61, 2006.

[65] IBM Internet Security Systems, "The Lifecycle of a Vulnerability," 2005, http://www.iss.net/documents/whitepapers/ISS_Vulnerability_Lifecycle_Whitepaper.pdf.

[66] US-CERT, "Multiple DNS implementations vulnerable to cache poisoning," 2008, http://www.kb.cert.org/vuls/id/800113.

[67] S. Frei, "Vulnerability Management," Master's thesis, Department of Management Technology and Economics - ETH Zurich, 2007.

[68] C. Miller, "The legitimate vulnerability market: Inside the secretive world of 0-day exploit sales," in *Workshop on the Economics of Information Security (WEIS 2007)*, 2007.

[69] R. Böhme, "Vulnerability Markets. What is the Economic Value of a Zero-Day Exploit?" in *Private Investigations (Proc. of 22nd Chaos Communication Congress)*, 2005.

[70] Symantec, "Report on the Underground Economy," 2008, http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf.

[71] T. Gregg Keizer, "Bug bounties not dangerous, 3Com claims," 2007, http://www.techworld.com/security/

news/index.cfm?newsid=9768.

[72] M. Blaze, "Is it harmful to discuss security vulnerabilities?" 2003, http://www.crypto.com/hobbs.html.

[73] S. A. Shepherd, "Vulnerability Disclosure," SANS InfoSec Reading Room - Threats/Vulnerabilities, 2003, http://www.sans.org/reading_room/whitepapers/threats/.

[74] J. T. Chambers and J. W. Thompson, "NIAC Vulnerability Disclosure Framework," Department of Homeland Security DHS, 2004.

[75] J. Moss, "Off at a Tangent – A discussion with Jeff Moss," *Computer Fraud & Security*, vol. 2008, no. 9, pp. 7 – 10, 2008.

[76] S. Christey and C. Wysopal, "Responsible Vulnerability Disclosure Process," 2002, http://tools.ietf.org/html/draft-christey-wysopal-vuln-disclosure-00.

[77] IBM Internet Security Systems - X-Force, "X-Force Disclosure Guidelines," http://documents.iss.net/literature/vulnerability_guidelines.pdf.

[78] F. Biancuzzi, "The Laws of Full Disclosure," SecurityFocus, 2008, http://www.securityfocus.com/columnists/466.

[79] MITRE, "Common Vulnerabilities and Exposures (CVE)," http://cve.mitre.org.

[80] ——, "MITRE Corporation," http://www.mitre.org.

[81] R. A. Martin, "Integrating your information security vulnerability management capabilities through industry

standards (CVE & OVAL)," vol. 2, pp. 1528–1533 vol.2, 5-8 Oct. 2003.

[82] MITRE, "CVE Candidate Numbering Authorities," http://cve.mitre.org/cve/cna.html.

[83] ——, "CVE Editorial Board," http://cve.mitre.org/community/board/index.html.

[84] NIST, "National Vulnerability Database (NVD)," http://nvd.nist.gov.

[85] FIRST, "Common Vulnerability Scoring System (CVSS)," http://www.first.org/cvss.

[86] Milw0rm, "Milw0rm Exploit Archive," http://www.milw0rm.com.

[87] Packetstorm, "Packetstorm Security," http://packetstormsecurity.org/exploits50.html.

[88] Securityvulns, "Computer Security Vulnerabilities," http://securityvulns.com/.

[89] D. Sornette, *Critical Phenomena in Natural Sciences: Chaos, Fractals, Selforganization and Disorder: Concepts and Tools (Springer Series in Synergetics)*. Springer, April 2006, ISBN-13: 978-3540308829.

[90] S. E. Page, *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press, January 2007.

[91] Gunter Ollmann, "Top-10 Vulnerable Vendors," http://blogs.iss.net/archive/Top10Vendors.html.

[92] OSVDB, "Open Source Vulnerability Database," http://www.osvdb.org.

[93] H.D. Moore, "The Metasploit Project," http://www.metasploit.com.

[94] C. Leita, M. Dacier, and G. Wicherski, "SGNET: a distributed infrastructure to handle zero-day exploits," Institut Eurecom, Tech. Rep. EURECOM+2164, 2007.

[95] Microsoft, "Windows Error Reporting," http://technet.microsoft.com/en-us/library/bb490841.aspx.

[96] G. Ollmann, "The evolution of commercial malware development kits and colour-by-numbers custom malware," *Computer Fraud & Security*, vol. 2008, no. 9.

[97] B. David, P. Pongsin, S. Dawn, and Z. Jiang, "Automatic Patch-Based Exploit Generation is Possible: Techniques and Implications," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, 2008, pp. 143–157.

[98] Microsoft, "Acknowledgment Policy for Microsoft Security Bulletins," 2000, http://www.microsoft.com/technet/security/bulletin/policy.mspx.

[99] R. Graham, "Disclosure ethics apply to BOTH parties," 2007, http://erratasec.blogspot.com/2007/01/disclosure-ethics-apply-to-both-parties.html.

[100] D. DeJean, "Windows XP: Going, going ... gone," 2008, http://www.computerworld.com/.

[101] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945, 1996, http://www.ietf.org/rfc/rfc1945.txt.

[102] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2068 (Proposed Standard), 1997.

[103] Mozilla Foundation, "User-Agent Definition," http://www.mozilla.org/build/revised-user-agent-strings.html.

[104] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Communications of the ACM, vol. 51, no. 1 (2008), pp. 107-113*, 2008.

[105] Jupitermedia Corporation, "TheCounter.com. Web Analytics," http://www.thecounter.com/stats/2008/June/browser.php.

[106] A. Dotzler, "Opera plans auto-update for version 10," 2008, http://weblogs.mozillazine.org/asa/archives/2008/12/opera_plans_aut.html.

[107] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, "The Ghost In The Browser - Analysis of Web-based Malware," in *Proceedings of HotBots 2007, Usenix*, 2007.

[108] Secunia, "Secunia Software Inspector Statistics (PSI)," 2008, http://secunia.com/software_inspector_statistics.

[109] B. Livingston, "Is IE 7 Really More Secure Than IE 6?" 2006, http://itmanagement.earthweb.com/columns/executive_tech/article.php/3639566.

[110] S. Hardmeier, "Microsoft Better Browsing - Internet Explorer 7 offers improved security and productivity," 2006, http://windowshelp.microsoft.com/.

[111] Adobe Inc., "Browser plug-in Market Shares," http://www.adobe.com/products/player_census/flashplayer/tech_breakdown.html.

[112] D. Sahal, "Foundations of Technometrics," *Technological Forecasting and Social Change*, vol. 27, no. 1, 1985.

[113] Carnegie Mellon University, "DARPA Establishes Computer Emergency Response Team," 1988, http://www.cert.org/about/1988press-rel.html.

[114] US-CERT, "US-CERT," http://www.us-cert.gov/aboutus.html.

[115] Bugtraq, "Bugtraq Security Mailing List," SecurityFocus, http://www.securityfocus.com/archive/1.

[116] Symantec, "Symantec," http://www.symantec.com.

[117] IBM Internet Security Systems - X-Force, "X-Force Advisory," http://www.iss.net.

[118] Secunia, "Vulnerability Intelligence Provider," http://www.secunia.com.

[119] John Cartwright, "Full Disclosure Mailing List," http://lists.grok.org.uk/full-disclosure-charter.html.

[120] FrSIRT, "French Security Incident Response Team," http://www.frsirt.com.

[121] SecurityTracker, "SecurityTracker," http://www.SecurityTracker.com.

[122] SecWatch, "SecurityWatch," http://www.secwatch.org.

[123] BlackHat, "BlackHat Technical Security Conference," http://www.blackhat.com.

[124] DefCon, "DefCon Security Conference," http://www.defcon.org.

[125] Open Security Foundation (OSF), http://www.opensecurityfoundation.org.

Curriculum Vitae

Stefan Frei received his diploma degree (MSc) from the Department of Electrical Engineering at ETH Zurich (Swiss Federal Institute of Technology) in 1995. As winner of a scholarship he wrote his master thesis at école nationale supérieure des télécommunications (ENST) in Paris. Thereafter, he joined InfoByte as a software/hardware design engineer. To fully exploit his interest in Internet technologies he started his own company in 1997, providing consulting on the development and security of e-commerce projects. End of 2000 he joined Internet Security Systems (ISS) as a member of the X-Force security assessment team London. He participated as technical lead in acquisition, delivery and execution of high profile, cutting edge attack-based consultancy services - ranging from classic penetration testing through to advanced Web application security assessments for an international high profile client base throughout EMEA. Stefan Frei rejoined academia in autumn 2004 for a Ph.D. research position in information security (Ph.D. defense in January 2009) and parallel post-graduate studies in management, technology and economics (Master of advanced studies (MAS), graduated in October 2007), both at ETH Zurich. His goal was to meet his security experience with academic research. He initiated and successfully directed the development of the content of a new networking security course at ETH Zurich. He has taught the course since 2006.

Acknowledgements

First I thank Prof. Dr. Bernhard Plattner for enabling my research and for the freedom to explore an interesting and interdisciplinary field of security research. I am particularly grateful to my family, especially my wife Suzanne and our daughter Chiara for their unconditional support and generosity. I am further indebted to Martin May and Thomas Dübendorfer for their continued support and introducing me into the world of scientific research. I want to specially thank Prof. Dr. Didier Sornette and Prof. Dr. Dr. Frank Schweitzer from MTEC for their collaboration and the excellent insight they provided me into complex systems. Furthermore my gratitude goes to Prof. Dr. Marc Dacier and Gunter Ollmann for their inspiring feedback and critics and for being my co-advisors. I also want to thank my friend Gallus Bammert for the many inspiring discussions that helped me decide to go for a Ph.D. after spending close to a decade in the industry.

I would like to thank my colleagues at ETH Zurich, in particular my office mates Daniela Brauckhoff and Bernhard Tellenbach and (in alphabetical order) Andreea Picu, Ariane Keller, Arno Wagner, Bernhard Distl, Brain Trammell, Daniel Sigg, David Hausheer, Dominik Schatzmann, Eduard Glatz, Elisa Boschi, Franck Legendre, Gabriel Popa, Georgios Parissidis, Ilias Raftopoulos, Jan Gerke, Kostas Katrinis, Lukas Ruf, Marc Rennhard, Marcel Baur, Mario Strasser, Martin Burkhart,

Merkourios Karaliopoulos, Placi Flury, Rainer Baumann, Simon Heimlicher, Theus Hossmann, Thomas Maillart, Thrasyvoulos Spyropoulos, Ulrich Fiedler, Vincent Lenders, and Xenofontas Dimitropoulos. My research would not have been possible without the active support of many partners in the industry. I want to thank:

Thomas Dübendorfer and his colleagues at Google Research for the fruitful collaboration and for providing me access to anonymized Web log data for my analysis. Wolfgang Kandeck and his team from Qualys for their support and for providing me anonymized data of their vulnerability scanning engines. Gunter Ollmann, Peter Allore, Luann Johnson, and the IBM X-Force team for their support and insight into cutting edge vulnerability research. Gregor Frey, Matthias Bossardt, Florian Widmer, and the whole IT Advisory team from KPMG Switzerland for their support of my patch management studies. Hannes Lubich from British Telecom for the many interesting and inspiring discussions.

Special thanks also to my colleagues Paul Beck and Urban Mäder who made our post-graduate studies in management, technology and economics at ETH Zurich an unforgettable experience, and Andrew Reichmuth for the professional insight into the fields of social engineering and police security services.

Last but not least I thank all others that influenced and supported the research of this Ph.D. thesis and that were not explicitly mentioned here.