Risk-Based Vulnerability Disclosure: Towards Optimal Policy

Andrew C. Dingman and Gianpaolo Russo
Indiana University
{adingman,russog}@indiana.edu

Abstract

Computing has become increasingly ubiquitous and embedded (as demonstrated by industrial control systems, in-vehicle systems, in-home care systems, and within the energy and transportation infrastructures). As a result, the issue of responsible vulnerability disclosure has returned to the fore. These new computing contexts require revisiting the nature of vulnerabilities and redefining responsible disclosure. The first goal of this work is to critique current disclosure practices. Based upon these critiques, grounded in the history of vulnerabilities, and informed by a series of expert interviews, we propose a model of risk-based responsible disclosure.

Research on vulnerability disclosure policy was an early focus in economics of security, particularly until 2006. That earlier research, however, reasonably assumed models of computers that were applicable to desktops, laptops, and servers. That is, there is a centralized source of patches, patching is possible in a very short time frame, patching is low cost, and the issue of physical harm need not be addressed. Current disagreements arise in part from the increasing diversity of both vulnerabilities and their potential impact. There are some clear lines. For example, it is not acceptable to disclose a vulnerability by implementing it and causing harm to victims. There are also well-known reasons for disclosure, specifically for creating incentives for vendors to patch and diffusing information to potential victims for their use in risk mitigation.

The trade-offs between transparency and confidentiality are increasingly complex. Responsible disclosure must be equitable: informing the marketplace, incentivizing software manufacturers to patch flaws, protecting vulnerable populations, and simultaneously minimizing the opportunities for malicious actors. To understand and resolve these challenges, we begin with the current state of vulnerability research. Stepping back provides a high-level historical perspective from the first identifiable vulnerability in a mass-produced device (beyond the canonical physical bugs in the first highly custom computers) to the Superfish malware in 2015. We describe extant models of disclosure, identifying the strengths and weaknesses of each of these. After that, we summarize factors previously used as vulnerability (and thus disclosure) metrics. These historical analyses and technical critiques are augmented by a series of interviews with technology and policy experts.

For the vast majority of vulnerabilities, the questions of public disclosure are not "if" but rather when and at what level of detail. We conclude that there is now no single optimal disclosure regime. Given this, we advocate for a model of disclosure grounded in risk-based analysis. Such an analysis should be complete and deterministic for a given context. We propose the factors necessary for such a systematic analysis. We then use well-known cases to test the framework and provide illustrative but practical examples.

This paper was prepared under the supervision of Prof. L. Jean Camp at Indiana University and submitted with her endorsement.

1 Methodology

The purpose of this work is to identify a socially optimal policy for vulnerability disclosure that can be utilized by organizations and individuals. It consists of two parts. The first consists of a foundation through the compilation and synthesis of research studies that cover many sides of the disclosure policy debate. The various mechanisms available for disclosure are reviewed and compared through available quantitative research.

Second, with this foundation, we interview experts with experience in vulnerability disclosure in the security field to create a sophisticated and discerning look at the nature of modern technologies, the effects of various disclosure behaviors, and what optimal policy needs to consider.

2 Overview

In the information age, intelligence on vulnerabilities within information systems is critically important. Everyday life, even in the hardest to reach places, is becoming increasingly entwined with the digital world. Consequently, the effects of vulnerabilities and the problems which they can create could become more hazardous. There are many factors to consider when deciding how to manage the disclosure of vulnerability information. For example, some factors which may be worthy of consideration are:

1. Where does the vulnerability occur?
2. What does it enable?
3. How easily can it be fixed?
4. Is it being actively exploited?

Naturally, various entities use vulnerabilities for different purposes. Vendors have an interest in keeping their consumers safe, therefore, they could use it to develop solutions which would protect as many users as possible. Agencies tasked with intelligence gathering could use vulnerabilities to exploit the information systems of their targets. Criminals use such vulnerabilites to further their schemes by infiltrating systems to gather information which can be monetized on black markets.[23] There are many interested and involved parties that an independent security researcher must balance when choosing a strategy to disclose the researched findings.

Additional questions then become added to the evaluation of a disclosure mechanism. What are the incentives posed by a given disclosure mechanism? What are the risks of disclosure? What are the incentives or likelihood for affected users to patch? Also, an effective mechanism needs to provide an incentive for white hat (non-malicious) experts to investigate and discover vulnerabilities.

2.1 History of Vulnerability Disclosure

In this section we introduce two early vulnerabilities and the associated responses. We also provide an initial overview of vulnerability response through the Common Vulnerability and Exposure project and Computer Emergency Response Team which was created in response due to one of these early events.

2.1.1 Therac-25

The case of the Therac-25 provides the first widespread vulnerability, which was initially not disclosed.[2] In fact, this first vulnerability can inform our discussions of ubiquitous systems. Therac-25 was a radiation treatment machine primarily used for cancer. It featured several different modes that provided different kinds of radiation. The machine had been derived from a previous version of a similar device, but it relied on software control in place of several safety systems from the previous model.

There were severe control system bugs resulting in six massive overdose accidents between 1985 and 1987. The system vendor was receiving reports of these incidents, but it did not share the information with other users nor identify and correct the flaws in a timely manner. Indeed, despite a lawsuit over severe radiation burns from the first known accident victim, medical service providers (i.e., users) were not notified of potential malfunctions until the doctors themselves discussed the failures, identifying them as systematic. Even in the face of regulatory action and a recall, the manufacturer continued to assert the impossibility that the Therac-25 could have malfunctioned to cause the injuries in a second and third case. When that third accident occurred, the operators of the machine were specifically told that no other incidents had occurred.[2]

Following a 1986 accident which resulted in the death of a patient, the manufacturer continued to deny the possibility of a problem, despite the aforementioned lawsuit and overdose incidents, and did not communicate that history to the users involved in the latest accident. A second similar accident at the same facility a few weeks later inspired the facility physicist to begin a more thorough investigation. Ultimately these two accidents resulted in regulatory action by the FDA including requirement of notice to users, as well as the creation of a user group to share data between Therac-25 customers. This process led to investigation and correction of several issues by the manufacturer.

Patching likely would have come along earlier had there been disclosure, because people would have pieced together that there were systemic problems that needed to be patched.[5] This expectation is supported by the fact that the vendor's response after an accident in 1987, when the user group was well established, included the announcement of a software update within about two weeks of the incident as well as reintroduction of a physical safety mechanism.[2]

2.1.2 Morris Worm

The Morris Worm, created by Robert Tappan Morris, is the second well-known vulnerability incident in widely used software. This illustrates scale and scope without physical harm. In this case, "disclosure" came in the form of an attack in November of 1988, which exploited a bug in the widely deployed Sendmail mail server to spread to "approximately 6,000" infected systems[1]. The 1989 Cornell Commission on Morris and the Worm suggests that the vulnerabilities exploited were already known to many members of the Unix community, although no formal disclosure mechanisms existed at the time.[1] It also suggests this may have been an early example of independent rediscovery by the author of the worm. Based on the broad impact of the Morris Worm, the Computer Incident Response Team (see section 2.1.4) was founded in 1988 to coordinate responses to future Internet security events.

2.1.3 Common Vulnerabilities and Exposures

The state of vulnerability discovery and disclosure has changed drastically in both scale and impact since the first two incidents. Rather than a small handful of vulnerabilities and incidents known by their names,

publicly known vulnerabilities are now tracked using MITRE CVE numbers. As of November 20, 2014, MITRE lists 8,813 CVE numbers issued in 2014. This number is higher than previous years, but consistent with a clear trend of growth from the 1,578 CVE numbers listed for 1999.[35]

2.1.4 Early CERT Policy

CERT/CC was created with the goal of improving responses to malware, in particular the Morris worm. Because of that concern, the original policy was to disclose vulnerabilities only to software vendors. The purpose of this policy was to enable vendors to resolve vulnerabilities before they become widely known, and thus prevent or reduce the impact of attacks using those vulnerabilities. Under that model, vulnerabilities were only disclosed to users and the public after a patch was available.

Under the initial policy, CERT/CC found that some parties who discovered vulnerabilities felt it necessary to provide public disclosure, to pressure vendors to address vulnerabilities rather than rely on secrecy of the information to protect their reputations.[17] To avoid circumstances where vulnerabilities were published before a vendor could respond with a fix, while satisfying the need for some pressure to produce it, CERT/CC policy was updated.

Current CERT/CC policy uses a delayed disclosure model in which security flaws are embargoed for distribution for a period after they are reported. The goal of such a policy is to allow vendors to issue patches or other mitigation tools prior to public announcement, in the hope that a fix will be available when disclosure occurs.[17] Details are, however, publicly disclosed after a defined period - usually 45 days - to increase the likelihood of timely vendor response.[17, 5]

3 DIFFERENT MECHANISMS FOR DISCLOSURE

There are many mechanisms for security researchers to disclose vulnerabilities. Each mechanism is subject to different economic influences, driven by different incentives and motivations, and raises different policy questions. In turn, each has its advantages and disadvantages, and has different outcomes for social welfare.

3.1 Computer Security Incident Response Teams (CSIRTS)

Computer Security Incident Response Teams work to collect, triage and disseminate information through collaboration and networking.

One of the oldest groups to do this in the United States is the CERT Division of the Software Engineering Institute (SEI) at Carnegie Mellon University. CERT originally manifested as the CERT Coordination Center (CERT/CC) in 1988 at the direction of the Defense Advanced Research Projects Agency (DARPA) in response to the first known computer malware, the Morris worm. The purpose of CERT was to quickly and effectively coordinate communication amongst experts in security emergencies in order to contain, mitigate, and prevent further incidents.

At the time there was not the industry of security researchers developing and monetizing bugs. Over time CERT became more widely used and less of a government organization, with much more industry participation and funding. Following up on CERT, there was interest in similar response organizations in other jurisdictions. CERT became a global clearinghouse, illustrating the value of having a national coordination center for vulnerability disclosure.

Third-party programs possess several advantages. Using a response team may provide a discoverer some insulation against a litigation onslaught from vendors. In addition, it is easier for a central response team to maintain working relationships with a broad selection of software vendors than for an individual discoverer to do so.[19] Personnel at a response team are also likely better experienced and equipped to understand the severity, scope, and difficulty of remediation associated with a given vulnerability than an individual discoverer.[19]

3.1.1 Open Code

Open source software, which is developed in public by groups of cooperating developers, poses an additional complication for disclosure. When the "vendor" for a piece of software is not a single individual or a formally defined group such as a corporation, non-public disclosure becomes difficult. In the case of the Linux kernel, for example, development is managed using a public mailing list (linux-kernel@vger.kernel.org) which is widely read and archived.[27] While disclosure to this list would accomplish the goal of reaching the developers of the Linux kernel, it would also be tantamount to public disclosure.

Multiple solutions to this issue exist. For some open source software, it is relatively easy to identify a single corporate entity that maintains stewardship of the project. In that case, such a company makes a logical choice to fill the vendor role in disclosure and patching. For example, if a vulnerability is discovered in the MariaDB database server, it would make sense to report the vulnerability to MariaDB Corporation Ab. Other projects lack an easily identified single corporate sponsor, but feature well-defined nonprofit governance. The Apache Software Foundation provides an example of such a governance organization, which could provide a suitable point of contact for non-public disclosure of vulnerabilities in the Apache HTTP Server project, Tomcat, and many other projects. Finally, for components with many stakeholders, there may be mechanisms such as Openwall's linux-distros list which allow restricted disclosure to verified representatives of several stakeholder organizations.[28]

3.2 Market-based Mechanisms

Recently, many disclosure mechanisms have arisen that are driven by market-based forces. The recent mood of a 'cyber arms race' means many parties are interested in getting exclusive rights to zero-day vulnerabilities and exploits. These parties include government agencies, who might purchase and stockpile the technology for espionage, defensive, or offensive purposes. Companies in the private sector develop their own defenses, services, or marketing. Criminals look to expand their portfolio of cybercrime. Players in the exploit buying and selling game range from small firms such as Vupen, Endgame, and Netragard to large defense contractors, such as Raytheon and Northrop Grumman.[36] Business is good. Adriel Desautels, the founder of Netragard, was quoted by Forbes saying the market has exploded with both the supply of zero-days and the demand for them shooting up. There are more sellers and more buyers with deep pockets entering the scene every year. A large part of what makes these markets possible is the influx of money starting from nation-state sources. An analysis of the US National Security Agency's black budget, made possible courtesy of the Edward Snowden leaks, showed that in the year 2013 alone, the NSA spent more than $25 million dollars to purchase vulnerabilities from vendors. Independent brokers operating as middlemen between developers and government agencies have recently said that business is so good that they don't have to touch deals less than five figures.[24] These kind of market forces allow firms like Vupen, a firm that specifically caters to government intelligence and law enforcement clients, to turn down decently sizable

offers from the software vendors in favor of a business model that keeps exploits private and only sold at premiums to the highest bidder.

3.2.1 Vendors

Software development companies have also developed mechanisms that aim to incentivize security researchers with rewards and bounties for reporting vulnerabilities. These 'bug bounties' can reward researchers anywhere from several hundred dollars to several thousand for the most sophisticated vulnerabilities. Some of the highest rewards come from contests such as that held by Google at the Vancouver Pwn2Own hackathon, where winner's could receive $60k for the details of their work.[37]

Full disclosure to the vendor, oftentimes with hope of receiving credit upon patch, is to be differentiated from full public disclosure. A hybrid model is the 'timed-disclosure' model, where the vendor is notified of the vulnerability and given the opportunity to patch, and then if no patch is released, the disclosure is made openly to the public.

3.2.2 Third-Party For-Profit

Another mechanism for vulnerability disclosure is found in programs such as HP TippingPoint's Zero Day Initiative (ZDI) and Verisign's iDefense Vulnerability Contributor Program. These programs work to provide defense security services to clients while simultaneously attempting to achieve a 'responsible disclosure' model. Both programs pay security researcher's for their work, and then communicate with the implicated vendor so a patch can be developed. In the down-time before a patch is released, the firms provide threat intelligence and stopgap protection to their clients.[38]

4 FACTORS USED TO ASSESS DISCLOSURE PRACTICES

In considering the optimal disclosure policy, we must consider the effect of disclosure policy on outcomes. Risk will depend on several factors, including likelihood of independent rediscovery, likelihood of exploitation if known, severity of harm from exploitation, and length of vulnerable periods both from disclosure to patch availability and from patch availability to widespread adoption. Since the length of exposure period is strongly influenced by both publication of vulnerability information and by vendor reaction to that information, the effect of vulnerability disclosure methods on vendor behavior is also highly relevant.

Ease of exploitation also appears anecdotally to have some effect on risk, as can be seen from such recent vulnerability discoveries as the Shellshock and Heartbleed vulnerabilities. Shellshock exploits are relatively easy to write[32, 34], requiring only a basic grasp of bash scripting to obtain control over an application's execution context. Useful Heartbleed exploits, on the other hand, require understanding the internal memory allocation within processes using the OpenSSL library, making large numbers of requests to obtain leaked portions of that memory space, and reassembling the leaked data into usable form. Initially, there was doubt in the security community as to whether exploitation was practical, although the debate was quickly resolved through a hacking challenge offered by CloudFlare.[33]

Vulnerability rediscovery has been addressed by other researchers, including Ozment in 2005. Ozment notes several difficulties with the available data on vulnerability rediscovery, but none the less provides evidence that the phenomenon is not unknown. He also proposes several possible models which could reasonably explain the phenomenon, including similarity of tools, limited search space, and commonality

in awareness of targets.[13] His evidence is derived from available information on rediscovery by benign actors. Although risk depends on rediscovery by malicious actors rather than by a second benign actor, it would appear reasonable to assume that malicious actors are affected by these factors as well. Certainly the search space is the same, and malicious actors have access to the same public information which influences awareness among benign practitioners. Given the existence of a professional class of computer criminals, it seems unreasonable to assume that they do not have access to the same tools through either legitimate or clandestine channels.

Aurora et. al. provide insight into the likelihood of exploitation. In their published data, it is trivial to conclude that vulnerabilities will be exploited if known. This applies even when a patch is available, although attacks are reduced in frequency both by time since disclosure and by patch availability. These data, together with case studies such as those presented by Arbaugh, Fithen, and McHugh, suggest that we can safely assume the probability of malicious use once disclosed is high. However, intensity of such activity can still result in a lower probability of attack across the population of vulnerable systems. These two sources also provide insight into the length of vulnerability periods and the intensity of malicious activity over those periods. Aurora et. al. also provide some insight into the effect of disclosure on vendor responsiveness.

For a given vulnerability, the severity of potential harm must also be considered. The spectrum of potential harm runs from information leaks specifically in systems not widely used for sensitive data through the potentially physical catastrophes of losing control of safety and control systems, particularly in industrial or transportation equipment. Even given the same ease of deployment and availability of information, more effort is warranted to respond to a vulnerability that could result in widespread harm or death than one which might be primarily a nuisance.

Control and other embedded systems provide the additional consideration that they are often difficult to upgrade. Such systems are often deployed with the expectation that they can be installed as "appliances" and require no maintenance. Users may be legitimately cautious about deploying updates, particularly in the case of systems where malfunctions could have catastrophic results.

5 DISCLOSURE PRACTICE OUTCOMES

5.1 Non-disclosure

Under a non-disclosure mechanism, the discoverer of a vulnerability does not disclose any information about the vulnerability to any other party. Provided that the discoverer is benign, this prevents malicious use unless the vulnerability is independently rediscovered. However, it also denies the vendor an opportunity to provide a timely fix for the vulnerability and denies users the opportunity to employ their own mitigation mechanisms, such as migrating to an alternate product or placing additional monitoring or access restrictions on the vulnerable component.

Non-disclosure simplifies the risk to that associated with malicious rediscovery. No mitigation is possible on the part of the vendor or users, because they are not aware of the problem. If no malicious party independently discovers the vulnerability, there is also no exposure period. There is evidence, however, that independent rediscovery does indeed occur in the community of security researchers.[13]

Ozment makes an argument that for some vulnerabilities, this may in fact be the economically optimal choice.[13] It is entirely possible that for low-impact vulnerabilities, or those which are difficult to remedy, the costs to patch would outweigh the risk of rediscovery and potential damage. In such a case, restricting

availability of the information may be the economically desirable course. Unfortunately, it is often difficult for the discoverer of a flaw to determine either the likelihood of independent discovery, the scope of potential damage, or the difficulty of deploying a remedy.[18] In addition we believe that response to less serious vulnerabilities may serve to develop and improve capacity to respond to more serious vulnerabilities in the future.

5.2 Disclosure to clients

Under this model, vulnerability information is disclosed by a security company to paying clients with an interest in the vulnerable software. For example, a company might pay a security firm to provide vulnerability information about software it deploys in the interest of applying its own mitigation or applying pressure to vendors to rectify the issue. The security firm might employ its own researchers to find such vulnerabilities, buy them from independent researchers, or both.

This model features a number of adverse effects. First, if a vendor or a given user does not choose to subscribe to a given security company's vulnerability data, it may be deprived of the opportunity to correct or mitigate the vulnerability. In addition, it is difficult or impossible to determine which potential recipients of vulnerability information would be considered malicious. One simple case would be a question of whether a given national intelligence organization constitutes a benign or malicious party, which might be disputed even within the nation it nominally serves. Screening the intentions of potential private purchasers is in some sense even more difficult, and may not be attempted by the firm at all. This opens the possibility that a client disclosure mechanism becomes a means not of protecting users of the vulnerable software, but of profiting from their eventual compromise.

According to our interviews, the time period between private and public disclosure of commercially disclosed vulnerabilities has been getting markedly shorter.[19] This may reduce concerns about the possibility that a vulnerability would be purchased by a malignant entity, since the window of time before the vulnerability is public and can therefore receive a mitigating response is shorter. None the less, if a vendor is not among the clients receiving disclosure, this mechanism extends the exposure period at least as far as the full disclosure model and potentially further.

5.3 Disclosure to tested clients

This model is similar to the previous, but it further restricts the audience for vulnerability information to paying clients who have been determined to be using software which contains the vulnerability. This model might be used, for example, in conjunction with penetration testing service.

The positive and negative aspects of this model are quite similar to those of the previous model as well. The opportunity for disclosure to a malicious party may be reduced by the requirement to make use of a vulnerable product before learning of its vulnerabilities. However, this may pose little impediment to a malicious party interested in particular target systems. Malicious actors may in fact be willing to pay as much or more for vulnerability information as people whose only interest is defense.

5.4 Disclosure to Vendors

A discoverer using this mechanism confidentially provides detailed information about a vulnerability to the vendor of the vulnerable software. On the surface, there are many potential benefits. The vendor has an opportunity to remedy the vulnerability, potentially before it is discovered and exploited by malicious

actors. The vendor may also have access to customer information which would allow dissemination of some vulnerability information to customers for their own mitigation efforts without making the information readily available to malicious parties. (Although it is impossible to eliminate the possibility that some customers are or could become malicious.)

Depending on a vendor's behavior, this mechanism can create outcomes similar to non-disclosure, since a vendor may choose to operate on the assumption that the vulnerability will not be independently discovered.[13, 14, 15] Under this assumption, a vendor may be disinclined to provide a patch which would implicitly acknowledge the vulnerability. The choice is not irrational given the evidence that vulnerability and breach disclosures can have an adverse effect on company valuation.[12, 47] Addressing software flaws also requires investment of development resources that might be more profitably used elsewhere. However, because the vendor is unlikely to bear the full cost of any compromise, this calculation does not necessarily represent a desirable outcome for all parties. Indeed, in the general case a vendor will take longer to patch than models indicate would be socially optimal.[15, 13] Actual behavior of vendors varies, in part based on both the competitive environment in which the vendor operates and whether the vendor uses a proprietary or open source development methodology.[5] Additionally, it can be difficult for an independent discoverer to identify a suitable avenue for confidential disclosure in the case of such things as open-source projects which lack a single corporate sponsor or proprietary software whose vendor has gone out of business.

5.5 Full Disclosure

Full disclosure attempts to address the difficulties of more limited disclosure by providing full details publicly, often including sample exploit code. One commonly stated reason for this is to incentivize vendors to provide a patch or other mitigation.[5] There is evidence that vendors experience economic loss from disclosed vulnerabilities[12], and also that disclosure timing does have an effect on the timing of patch release.[5, 14, 15] This suggests that disclosure can help reduce the degree to which impact on customers is treated as an externality by software vendors.

One significant detriment to this mechanism is the time required for response. Once information about a vulnerability is widely available, development of useful exploit code may be significantly faster than deployment of a remediation. In many cases, malicious actors can accept significant failure rates for automated exploits, reducing the time required to convert knowledge of a vulnerability into exploit code. Vendors and users, on the other hand, face significant obstacles in ensuring that a fix for one vulnerable system does not result in loss of other required functionality. Without such care, applying a patch could potentially cause more damage than it forestalls. Even in the best of cases, a vendor requires time to develop a patch, test it internally, and distribute it to customers. In many cases, additional time will be required for customers to perform their own quality and interoperability testing on the fix. In some cases, such as software embedded in appliances, control systems, and the like, there may be no realistic possibility of patching all affected systems. These factors result in a period of time after a full disclosure event in which benefits accrue primarily to malicious users of the vulnerability.

5.6 Disclosure to Response Teams

To address these issues the information security community has developed industry response teams, such as US-CERT and the CERT/CC. These organizations can accept vulnerability reports from discoverers, and coordinate disclosure to vendors and the public. In general, they operate by first disclosing details of a

vulnerability to the relevant vendor[17], or through mechanisms such as the "linux-distros" mailing list which are restricted to verified contacts for established open-source projects.

To maintain the desirable pressure to remediate the issue offered by full disclosure, such organizations typically disclose information about the vulnerability publicly after a period of time, regardless of whether the vendor has supplied a fix.[17, 19] In such cases, they typically provide descriptions of the general nature of the vulnerability and any known protective actions available to users of the affected system, but do not provide example exploit code and may not provide sufficient technical details to develop an exploit.

Another advantage of this method is that responses can be modulated through the judgment of domain experts. These experts may be better able to assess factors such as the severity of potential damage, ease of exploit automation, and difficulty of developing and deploying a fix. Although a vendor might also be able to make these assessments, there is an additional advantage in a separate response team which can consider the risks to customers in addition to the cost of a fix to the vendor.

6 Law To Consider When Forming Policy

A jurisdiction's law dictates what one is and is not allowed to do, and as such, interacts with and can constrain disclosure action. Many a hobbyist or researcher has discovered a vulnerability, and their understanding of what they are allowed to do with the information has created problems.[18, 20]

The body of law that needs to be considered varies widely with the categorization of the vulnerability and the systems affected. Information regarding financial systems, national security systems, and health information systems have distinct laws dealing with related actions. Criminal law can also apply to how the researcher came into contact with the information, how it was tested, and how they go about releasing it. Likewise, civil law needs to be considered for these actions, as well, from copyright and trade secrets, to damaging a brand, to contract violation and tortious interference.

When considering vulnerability information and its relation to national defense, US federal criminal law has the potential to be very strict regarding the communication and retention of such information. Section 793(e) of the United States Code Annotated criminalizes the communication of information related to the national defense, or even information which the unauthorized possessor has reason to believe could damage the US or be used to the advantage of a foreign nation.[21] This law also makes criminal the willful retention of that information, and the failure to deliver it to an entitled officer or employee of the US. This can be far reaching as many consumer software products are also running on government computer systems as well. A flaw in a major Internet browser from the private sector may seem to have far reaching implications for the general population of end users and the companies exposed, but how widely is the software used in government agencies? Without access to significant information about classified systems, a vulnerability discoverer is not in a good position to assess the scope of effects that a disclosure would have, which leads to the fact that they lack the knowledge to decide how the disclosure could be used.[18] This type of legal policy asks for vulnerability disclosure to be aware of the nature of the information at hand, and to take specific actions corresponding to this nature.

In the US, almost every state has its own law requiring that a company notify consumers in the event of a breach of personally identifiable information.[31] That notification is required is more or less significant depending on requirements for form, timeliness, and content of the notice. In an SEC guidance on disclosure action, "While registrants should provide disclosure tailored to their particular circumstances and avoid generic 'boilerplate' disclosure, we reiterate that the federal securities laws do not require disclosure that

itself would compromise a registrant's cybersecurity. Instead, registrants should provide sufficient disclosure to allow investors to appreciate the nature of the risks faced by the particular registrant in a manner that would not have that consequence."[16] As a policy guideline, the treatment of cybersecurity threats in these industries asks for discretion to be used in the actual disclosure. A disclosure should not be so explicit as to compromise or introduce further risk to a system. As of this writing, S.754[48] attempts to improve cybersecurity by increasing disclosure among trusted parties. It also removes liability upon disclosure of an event or vulnerability.

7 Assessing Disclosure Risk

The risks posed by a given vulnerability depend on a number of factors relating broadly to the severity of harm which can be caused by an exploit, including the scope of vulnerable systems, the cost of mounting an attack, and the available mitigation mechanisms. **Severity** may run from minor annoyance to financial loss, physical damage, and even death. Greater severity directly contributes to the risk associated with a vulnerability. Provided no wild exploit is known to exist, greater severity may argue for a longer non-disclosure period to allow mitigation options to be developed. However, it also argues for greater pressure on vendors to supply mitigation, which can be supplied by a deadline for full disclosure regardless of mitigation status. Greater severity, however, may warrant a longer time period from notification to disclosure.

Scope might range from a single bespoke device to millions of deployed systems subject to a vulnerability. Like severity, a larger scope increases the total risk associated with the vulnerability and its disclosure because potential damages are multiplied by the number of potential victims. When scope is extremely small, limited disclosure to users ahead of full vendor response may be practical. In general, greater scope will mean greater risk, and again may argue for a longer time period between notification and disclosure.

Cost of attack reflects the difficulty of developing an exploit for a vulnerability as well as the difficulty of delivering the exploit to a target system. For example, the recent Shell Shock vulnerability had very low cost. It was simple to understand the problem and write suitable exploit code, and that code could be delivered to many vulnerable systems over the Internet using commonly exposed protocols such as HTTP and SSH. Even complex vulnerabilities are frequently automated and commoditized, so that factor rarely contributes significantly to cost. Some vulnerabilities, however, may require physical proximity, as might be the case with a flaw in a Bluetooth stack or USB host firmware. Exploits for these types of vulnerabilities would require an attacker to already have control of a device within, at most, a few hundred meters of the target device, or to convince a target user to insert an infected device into a target system. Although these are certainly not insurmountable obstacles, they would raise the cost to an attacker of taking advantage of these types of vulnerabilities.

Mitigation mechanisms are more complex and break down along several dimensions. One is the **ease of creating an update** or other modification to the vulnerable product to address the problem. In a software context this is generally referred to as patching, but it could also involve hardware modifications such as the changes to switches and addition of hardware safety devices in the Therac-25. For some vulnerabilities, the fix is relatively straightforward and easy to implement. For example, the recent Heartbleed vulnerability was a relatively straightforward matter of failing to validate inputs. Modifying the code was not excessively difficult once the vulnerability had been identified, and the change in behavior by definition could not interfere with a non-malicious peer. More difficult, the original SSH protocol had a fundamental design flaw which required an incompatible replacement protocol.

Related to ease of developing a patch is the **ease of distributing and applying a patch** once it has been created. Returning to the Heartbleed example, although patch development was relatively straightforward, distributing the patch to all affected systems is much less so. Although typical web and internet services are managed by staff capable of applying software updates, many "appliance" type devices such as printers, routers, and smart home devices may have neither a straightforward update mechanism nor operators who are likely to apply updates even when available. Modification of some systems, such as implanted medical devices, may be functionally impossible.

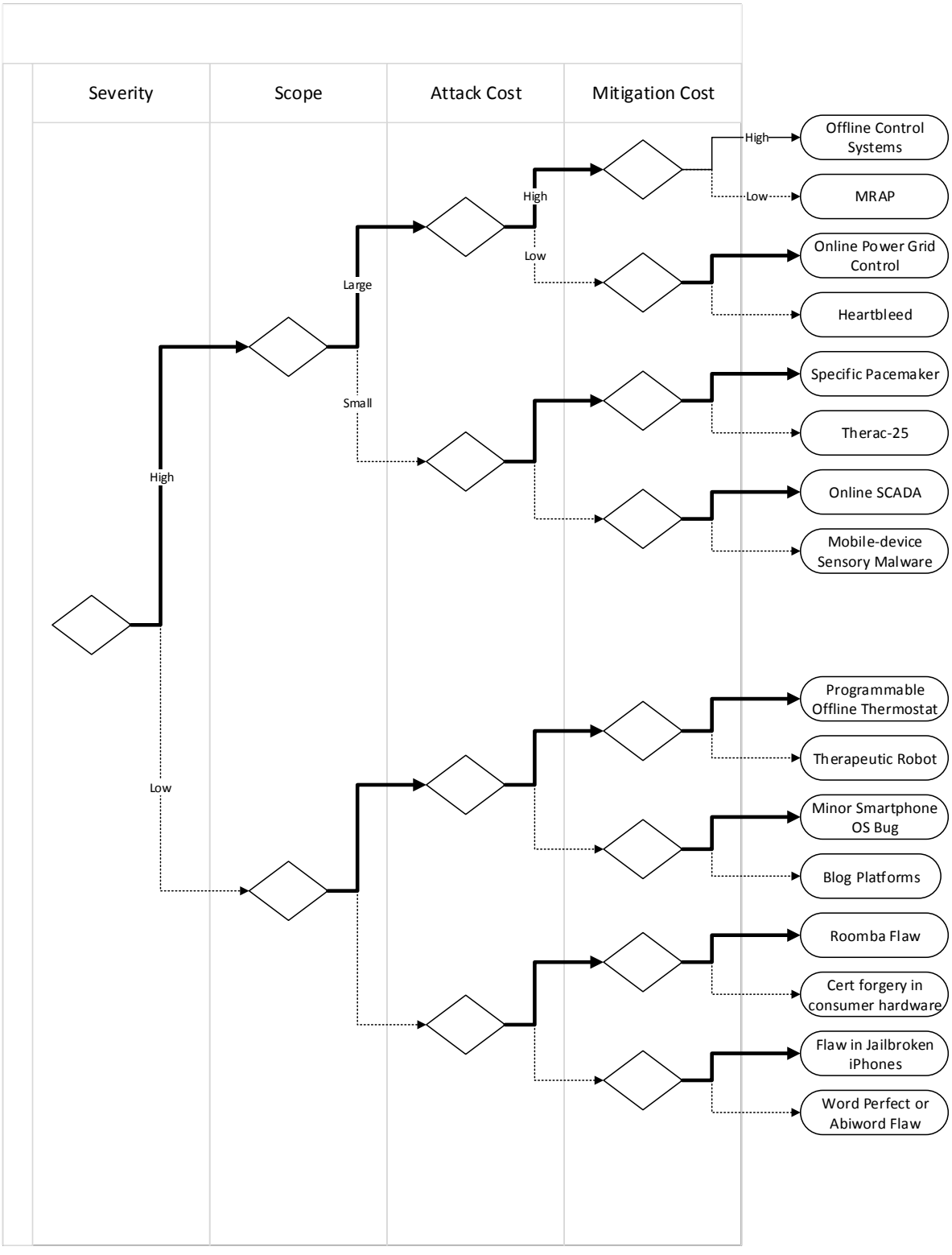
Beyond patching, other dimensions of user control may offer **viable mitigation options**. Available options vary depending on the degree of control the user community has over whether and how they use a vulnerable system, and may vary between users of the same system. At one extreme of choice, a user may be able to simply choose not to use a vulnerable system with little or no adverse impact. More commonly, a vulnerability might exist in a particular feature which many users could disable with little impact, or an alternative may exist with low acquisition and switching costs. In the case of devices which embed computing devices, it may be possible to impose network-level access controls which restrict access to a vulnerable interface without significant impact on the intended use of the device. For example, a security camera on a residential front porch might be restricted from communicating with the Internet entirely if the intended use is local observation before answering a knock at the door.

Given the **possibility of independent rediscovery** and the expectation that once known, a vulnerability will be exploited, available mitigation should weigh in favor of earlier and more complete disclosure. This maximizes the opportunity for the exposed user population to apply the available mitigation before exploits become widespread. This is highly intuitive for the simplest case; it is not hard to see that if a minor, easily distributed and applied patch already exists, the vendor and users are both best served by making that fix available. Naturally this extends to cases where other forms of mitigation are expected to be broadly applicable. As the proportion of the vulnerable population to which a viable mitigation is available shrinks, the value of early disclosure falls as well.

8 Disclosure Framework

Using the risk factors outlined above, we propose a process for determining an appropriate disclosure model for a given vulnerability. Each of the major risk factors above - severity, scope, attack cost, and mitigation cost - must be assessed for the vulnerability in question. Our decision tree illustrates interactions between the dimensions on a boolean scale. Real world analysis will require a wider range of values for each dimension to provide an adequately nuanced view of the risks and corresponding desirable disclosure practices.

Figure 8.1: Risk Categorization



8.1 Sample Vulnerabilities

Figure 8.1 shows the range of possible options under our proposed disclosure algorithm. Here, we discuss each item and classification. We close with a summary in Table 1.

8.1.1 Offline Control Systems

Cost of access to vulnerable offline SCADA systems is exemplified by the Stuxnet worm, where an advanced delivery system was developed simply to obtain the necessary access to compromise target control systems. Despite the narrow use in that case, other SCADA systems can be very widely deployed and often are extremely difficult to patch, especially in the case of offline or air-gapped devices which may require a physical visit or even outright replacement of device components to apply updates. Depending on the devices controlled by the vulnerable SCADA system, severity could include significant property damage, as in the case of the ruined centrifuges, and even deaths. Scope is potentially quite wide as well, given the large numbers of people who could be impacted by an attack on aircraft or industrial chemical control systems.

For this type of vulnerability we advocate a CSIRT model. The current model could be improved with an industry-specific response team, and inclusion of relevant regulatory agencies. Although high attack cost might delay an attacker, the other factors warrant care, delay, and limited disclosure until mitigation efforts can be completed.

8.1.2 MRAP

There is evidence of groups targeting military computer assets protected by "air gaps".[50] Targets such as the computer systems in MRAP (Mine Resistant Ambush Protected) vehicles could present high severity due to their military nature. Likewise the scope would be large due to the large numbers of such vehicles in use. Attack cost to any target behind an air gap is high. Unlike the SCADA systems described above, though, MRAPs are manned military vehicles regularly visited by professional users. Because of that, mitigation cost for a known vulnerability should be much lower because the organization infrastructure for identifying the vulnerable installations and applying maintenance measures is already in place and covers the entire installed base.

Low mitigation cost should allow rapid response. None the less, late publication of the vulnerability is advisable to allow ample time for creating and applying a remedy.

8.1.3 Online Power Grid Control

A high severity, large scope, difficult mitigation, low attack cost vulnerability could exist in online "smart" power grid control systems. The severity of such a vulnerability is high, since with power control systems an attacker can potentially damage very expensive equipment, start fires, or cause widespread power outages. The scope is large because the electrical grids are huge and consist of almost innumerable devices, while the attack cost would be low because these are network-facing devices. In this scenario, widespread disclosure might never be considered, or at least not until after mitigation is in place. Relating to critical infrastructure, the user community is known, so perhaps they can be approached quietly to create a situation where you get most of the benefits of wide disclosure while limiting the likelihood of the vulnerability becoming known to potential malicious actors.

The combination of low attack cost with high severity, scope, and mitigation cost make this type of vulnerability particularly attractive to hostile parties. Because of this combination, we recommend closed

disclosure to known defenders only. The discoverer, manufacturer, and any regulatory agencies should coordinate to develop remediation options, identify vulnerable users and systems, and disclose to them as needed to facilitate mitigation.

8.1.4 Heartbleed

The recent Heartbleed vulnerability is an example of a high severity, large scope vulnerability with low costs for both attack and mitigation, at least for most users. An SSL library is naturally expected to be used to protect sensitive data, including private financial and medical data. Clearly a vulnerability which can be used to compromise the SSL keys and any other data which may be in the memory space of a client process is severe. Similarly, OpenSSL is sufficiently widespread that there can be little doubt as to the large scope. Attack cost is negligible, as systems using SSL are frequently expected to be available remotely.

Mitigation cost, in this case, varied significantly. The patch was not particularly difficult to develop nor disruptive to apply. It was somewhat elevated, however, by the fact that the developers have little to no control over when downstream software adopts the fixed version or when users deploy it. Most vendors of desktop and server software made the updates available promptly. However, because of the extremely large scope, mitigation for some users was much more difficult. Appliance style devices, such as printers and routers, commonly embed OpenSSL in their firmware, which is usually more difficult to patch than typical server and desktop systems. Some of these devices, such as Cisco routers or work group printers, tend to be professionally managed and thus likely to have updates developed and applied. Vulnerable home equipment, on the other hand, will likely not be updated even when updates are available.

The combination of wide scope with low attack and mitigation costs in this case argued for public disclosure, particularly once a patch was available. Unlike our other high severity examples, in this case narrower disclosure to vulnerable users first would not have been possible, both because of the scope and because the OpenSSL project is not in a position to identify and notify users.

8.1.5 Specific Pacemaker

A pacemaker provides a straightforward example of a device in which security flaws present a clearly high severity, as such vulnerabilities could threaten the continued operation of a vital organ. A vulnerability in a particular model of pacemaker, however, would usually be small in scope due to the small number of individuals using that device. Attack cost would generally be high, as these types of devices are not remotely accessible. Mitigation cost would be high for similar reasons; indeed for the vast majority of patients it would likely not be worth the associated risks.

For this type of vulnerability, we advocate disclosure to the vendor followed by publication but without sufficient detail for an attacker to exploit the vulnerability. The small scope and high attack cost make it unlikely that an attacker would choose this method of attack to harm a target, particularly when they must first rediscover the vulnerability with incomplete guidance. Balancing that small net risk, this method preserves the vendor's incentives to provide what remediation is possible and guard against such flaws in the future. The experience of Therac-25 calls for disclosure, but the severity calls for both a delay and a lack of detail. The calculation of the delay must balance the risk that additional people are implanted with the pacemaker against the risk to those already implanted. Eventual disclosure is important to ensure manufacturer incentive.

8.1.6 Therac-25

Cost of access to Therac systems was high, because interaction with the machines required physical presence in the hospital. Additionally, the number of people qualified to work with the vulnerable systems was minimal, making it difficult for a malicious actor to obtain unfettered access even once present on site. Scope was small because of the small number of Therac devices built. That also eased mitigation, since there were few users to whom a fix needed to be distributed once it was developed. In addition, once the specific problems with the user interface were understood, it was possible for operators to mitigate the risks by taking care to avoid actions which might trigger misbehavior in the device. Even without that knowledge, it was possible to simply stop using the devices until more was known. Given that patients suffered severe injury and death as a result of the bugs, however, it was clearly a severe vulnerability.

In this case, we believe immediate, full disclosure would have been the best option. Mitigation was immediately available to any patient or hospital simply by discontinuing use of the machines until more was known. Full disclosure would almost certainly have saved lives and likely would have hastened a fix which could allow the machines to serve their intended function with acceptable safety margins.

8.1.7 Online SCADA

Remotely accessible SCADA systems used for process control in a relatively small industry that deals with potentially toxic chemical processes would have high severity due to physical safety risks, but a small scope. Once a vulnerability was known, network access would make the attack relatively cheap. However, such devices might very well require physical visits to apply updates, if not replacement of custom ROM or FPGA chips, making mitigation cost quite high.

Here, as with the first example, we advocate for a CSIRT with industry-specific expertise and communication with relevant regulatory bodies such as the EPA. That is, delayed but certain disclosure, yet with limited detail.

8.1.8 Mobile Device Sensory Malware

An illustration of this case could be a vulnerability in a smart phone application that allows exfiltration of sensitive data, such as private photo, video, and location information, giving it high severity, but which is not widely used. Attack cost would be low due to connectivity of the device. However, smart phone applications are generally easy to patch, and the patches are easily distributed through the app stores used for initial distribution. In the case a patch was not forthcoming, users would also have the option to simply remove the vulnerable application. (This therefore excludes widely used social networking apps with high social switching costs.)

For this type of vulnerability, we would recommend vendor notification followed by somewhat delayed publication to allow for patches to be developed and pushed. Full disclosure could serve future security research and is recommended.

8.1.9 Programmable Offline Thermostat

HVAC systems rarely have the capacity to heat or cool to life-threatening levels, rendering malicious control relatively low severity. However, they are widely enough used to have a potentially large scope. Attack cost for a typical non-networked thermostat would be high due to the requirement of physical proximity.

Mitigation costs would also be high due to the same requirement, as well as the difficulty of getting end users to apply the patch.

Harm in this case is sufficiently minor and unlikely. We believe immediate, full publication to be an acceptable course of action.

8.1.10 Therapeutic Robot

A device such as the PARO therapeutic robot might offer low severity combined with large scope and high attack and mitigation costs. Barring extreme cases, it is unlikely to do worse than fail to provide the comforting services intended. If popular, such a device might have wide scope, but would likely be difficult to attack because there would be little need for connectivity. That combined with the target audience would make it unlikely that any patch would be applied, raising mitigation cost significantly.

Harm in this case is sufficiently minor and unlikely that we believe open publication to be an acceptable course of action.

8.1.11 Minor Smart Phone Platform Vulnerability

A low severity, large scope, difficult mitigation, low attack cost vulnerability could be one that exists in the Android core system, can be triggered over the internet, and that causes trivial glitches for smart phone users, such as resetting background images or not saving settings. The scope is massive because Android devices are very widespread. Mitigation is very difficult because as this is a core system component rather than an app. Because of the distribution model for Android, the main vendor (Google) can't push the update, and phone companies push updates very slowly and generally need to rely on the equipment manufacturer, who in turn has little incentive to develop and test patches for equipment no longer being built for sale. In addition, cell phone networks are unlikely to push an update until they have done their own testing due to concerns about network stability.

For this type of vulnerability, we would advocate a vendor-first disclosure, followed by publication without details usable in an attack.

8.1.12 Wordpress or Xbox Minor Flaw

A typical low severity Wordpress exploit does not involve privilege escalation, and in most cases the platform does not come into contact with sensitive information. With the publishing platform, any information involved is meant to be shared, and with the video game system, most of the data is game save files and in-game assets. Altogether, severity for such flaws is typically fairly low. Scope, however is large because these are popular, widely used platforms. Both are network connected, providing relatively easy attack vectors, but also relatively easily patched, which gives low cost for both attack and defense.

For this type of case, it would be considerate to provide notice to the steward or vendor and somewhat delayed disclosure. However, open publication is an acceptable course of action.

8.1.13 Roomba Flaw

The potential for malicious use of a small floor cleaning robot is limited, and while popular, the scope for attack is also limited to relatively few households. However, attack cost would be high since there is no significant remote access capability. Mitigation cost would also be high, due to limited ability to distribute updated software to the devices.

Disclosure to the vendor followed by delayed, generic publication of vulnerability information would be ideal. The only practical mitigation in this scenario is to address the flaw in units shipped after the vulnerability is known. Patching the installed base is impractical, so vulnerable devices will remain in the wild for considerable time.

8.1.14 Certificate Forgery in Consumer Hardware

As an example for this category we propose a code-signing certificate employing a weak hash algorithm subject to FLAME-style collision attacks. However, in this scenario, the certificate is trusted only for updates to a consumer device which collects no sensitive data, such as personal outdoor weather monitoring hardware. Severity of such a breach would be low, with scope limited to a likely small installed base. Although the cost of attack would be high, since even weak hashes would require substantial computing resources to break, mitigation for such a device could take the form of a simple update that revokes and replaces the weak certificate.

The combination of low severity, a small scope, and high attack cost makes this a low risk scenario in which open publication is acceptable. Again, courtesy to the vendor might suggest a short delay between disclosure to the vendor and public disclosure.

8.1.15 Application Flaw Specific to Jailbroken iPhone

To illustrate this category, suppose a vulnerability only present in jailbroken iPhones using an application that is not hugely widespread. This vulnerability can be used by malware to exfiltrate sensitive data from its victims. There is a low mitigation cost for this vulnerability, because if the app is not in widespread use, users have the mitigation option of simply not using the app. This therefore excludes widely spread social networking apps which users would have social consequences for not using. This vulnerability is characterized by a low cost to attack: the difficulty of writing the initial exploit code is not a huge cost because it will eventually become automated and commoditized, and readily available to script kiddies.

In this case immediate publication is essentially the only option. The relevant vendor is unlikely to develop a mitigation for this vulnerability, unable to push it to vulnerable users, and generally not trusted by the vulnerable community. Thus vendor disclosure has essentially no utility, but open publication can allow users to protect themselves.

8.1.16 Word Perfect or AbiWord Flaw

This class of attack is typified by a crash or code execution bug on a non-mainstream word processor application. This bug does not have any foreseeable larger effects, such as privilege escalation. At worst it results in the takeover of one unprivileged user's account. Either is a desktop computer application that can be patched relatively straightforwardly. Here, rapid disclosure with limited time for vendor response is desirable.

Since the vulnerability should be easy to fix, vendors should not require much time, and they have an incentive since we know that vendors take a market hit for failing to patch. Even in this low risk case, some delay is desirable because with anything it is better to have a patch available before the information is made open to potential malicious attackers. Without disclosure, history suggests that no patch may be forthcoming.[5, 10]

Table 1: Risk Dimensions

	Severity	Scope	Attack Cost	Mitigation cost	High-level Summary
1	High	Large	High	High	Immediate limited disclosure; delayed, generic publication
2	High	Large	High	Low	Immediate limited disclosure; delayed full publication
3	High	Large	Low	High	Affected users only
4	High	Large	Low	Low	Immediate vendor disclosure; slightly delayed full publication
5	High	Small	High	High	Immediate vendor disclosure; delayed generic publication
6	High	Small	High	Low	Immediate full publication
7	High	Small	Low	High	Immediate vendor disclosure; delayed generic publication
8	High	Small	Low	Low	Immediate vendor disclosure; delayed generic publication
9	Low	Large	High	High	Immediate full publication
10	Low	Large	High	Low	Immediate full publication
11	Low	Large	Low	High	Immediate vendor disclosure; delayed generic publication
12	Low	Large	Low	Low	Immediate full publication
13	Low	Small	High	High	Immediate vendor disclosure; delayed generic publication
14	Low	Small	High	Low	Immediate full publication
15	Low	Small	Low	High	Immediate full publication
16	Low	Small	Low	Low	Immediate vendor disclosure; delayed generic publication

8.2 Closing

For most vulnerabilities, some variation of the responsible disclosure model is desirable. Recent developments have not changed the fundamental fact that the threat of disclosure provides one of the few available mechanisms by which software providers can be pressured to develop patches or other risk mitigation tools. It is also still the case that even with the best of intentions and conditions, a vendor may not be able to provide a tested, safe patch or other response that can be distributed and applied by their user base with the same speed that an attacker can exhibit in developing an exploit. Thus, in most cases, it is still desirable to provide a window of time between alerting the vendor of a vulnerable component and alerting the user community and/or the world at large. Exactly how much time, and who should be notified, however, varies with the risk characteristics described in the previous section.

8.3 Policy Summary

Each of the four dimensions of risk must be considered when choosing a disclosure strategy. As a rule, higher attack cost will argue in favor of earlier and wider distribution of vulnerability information. Holding other factors constant, greater cost for the attacker reduces the risk that vulnerability information will be used in an attack, while disclosure continues to provide an incentive for the developer of the vulnerable system to respond. Disclosure is also a prerequisite for applying any end user mitigation strategies.

Wider scope will tend to indicate a longer period between disclosure to the developer and disclosure to the user community or the public. This reduces the likelihood that the vulnerability is disclosed while it is still impossible for large numbers of vulnerable users to address their vulnerable status. Scope also influences who can be notified. With a smaller population of potentially vulnerable systems, it may be possible to disclose vulnerability details only to known users, who may be able to make use of the information defensively while having an acceptably low risk that one or more will use the information to mount an attack. This would be

especially beneficial in the case where a vendor may not yet have a full solution to the issue but there are known mitigation options that users can implement in the interim. However, when the scope is broad, as might be the case with such widely used software as the currently dominant word processor or a widely used web publishing platform, any attempt to notify only users would quickly result in the information becoming public.

Higher mitigation costs again argue for longer periods between disclosure to the vendor and wider disclosure. The greater effort required to mitigate the vulnerability would, in general, be expected to take greater time to accomplish. In the case of a scope small enough to permit disclosure to users ahead of the general public, it may also indicate a longer period between those steps. This is particularly desirable if part of the mitigation cost comes in the form of difficult patch deployment, such as might be the case for a vulnerability in field-deployed control or monitoring devices which may require a physical visit to update or replace the vulnerable component.

Greater severity, also, should result in a longer time between disclosure to the vendor and wider disclosure. The more damage could be caused by a malicious actor with knowledge of the vulnerability, the more desirable it is to allow time for a complete fix to be developed and delivered before the information becomes public.

9 Hazard Solution

Intellectual property, computer systems, networks, software all effectively have an owner. Even open code generally has identifiable stewards, be they commercial entities, non-profit organizations, or well-known individuals. (e.g., Red Hat Software for projects like the 389-ds directory server and oVirt; the Apache Software Foundation for Tomcat, Struts, and the Apache HTTP Server). If and when a problem occurs, the difficulty arises in finding a responsible way to convey the problem to the owner, whether the owner is a vendor, a government, an airline, a hospital, or a major corporation. In today's society, when a researcher discovers a vulnerability, unless otherwise under contract or legal restriction, a decision must be made regarding the disclosure of the vulnerability. However, the discovery of a vulnerability, in itself, does not necessarily place the responsibility of its revelation upon the researcher.

To disclose a vulnerability with details sufficient to enable an attack in order to force vendors to put a patch in place is an ethically suspect strategy. It makes the significant presumption that an individual or team of researchers can know all the constraints and circumstances under which a vendor is operating, and all of the customers that would be affected. This can be likened to a whistle blower's individual moral judgment that closely guarded national secrets should be disclosed to the entire world in order to instigate change. Such a singular individual action would have to presume that the individual who makes the disclosure is able to see all the pieces in play, knows the far-reaching effects of every program, how many people are affected and in what way, knows how the disclosing of very sensitive information will in turn affect those lives, and that they are in a morally superior position to make the judgment of what is best for those affected.

Less detailed disclosures by researchers, Koscher et all[49], can serve to notify vendors without providing opportunities to attack. Note that although that attack was of wide scope and had high mitigation costs, it also had a high cost of attack. A detailed disclosure would have risked severe attack.

Part of what makes disclosure decisions so complicated is that there are many situations where systems can not be easily patched, either due to regulation or location. Aircraft and flight control systems are an example of this. The sector is heavily regulated by the FAA, and these systems, as with many other safety critical systems, can take years to have changes authorized due to the extensive testing and verification

that needs to occur. These difficulties can be seen again and again in regulated sectors relating to critical infrastructure all the way down to important medical equipment. Disclosing a vulnerability in these systems in turn makes all of the systems vulnerable without a way to fix them, and endangers anyone using those systems.

Additional moral hazard is introduced when business entities take disclosure matters upon themselves. Where there is market competition, as between Google, Apple, and Microsoft, disclosing vulnerabilities and providing exploit details and code to the public can have significant external effects on reputation and market position.[12, 45, 46]

There certainly is no one-size-fits-all answer, but there are strategies that could be realized to maximize responsibility by mitigating these judgment risks. The first step is to establish a set of clear best practices for vulnerability researchers, vendors, and practitioners. The failure to follow this due diligence and best practice would clarify when liability or other sanction should apply.

Such a mechanism would resemble a coalition of sorts that could examine and review vulnerabilities from a holistic perspective. The coalition should adequately represent legal and technical expertise, as well as have multiple representatives from within affected sectors. This kind of joint-review organization would represent interests, expertise, and perspectives across a wide spectrum that come together for this common cause. Various sectoral CSIRTs could serve this purpose due to domain expertise.

Prior to disclosure, a researcher would submit their vulnerability research through this mechanism in order for the risk-based assessment to be made amongst a balanced group. Following review, the coalition could then function as a broker, managing the disclosure practices that correspond to the risk level and forwarding any remuneration to the researcher, or allow the researcher to manage disclosure under the coalitions recommendation. With the advent of so many financial and reputational incentives vying for the attention of this kind of security research, the adoption of such responsible mechanism need not interfere with these market forces. Specifically, a party such as ZDI could provide the vulnerability to the CSIRT along with its own customers.

Indeed, remuneration for research coming from the vendors or vendor groups themselves is a positive practice, as it can balance the draw to engage in underground markets and result in the research landing in the hands of those most capable of patching the flaw. Criminals and other groups will continue to have the incentive to stockpile vulnerabilities for their various machinations.[44] Yet a single model for disclosure is not the answer.

References

[1] T. Eisenberg et al., “The Cornell Commission on Morris and the Worm,” Comm. ACM , vol. 32, no. 6, June 1989, pp. 706–710

[2] Leveson, N.G.; Turner, C.S., "An investigation of the Therac-25 accidents," Computer , vol.26, no.7, pp.18,41, July 1993

[3] Mohammad S. Rahman Karthik Kannan, Mohit Tawarmalani, Purdue University, The Countervailing Incentive of Restricted Patch Distribution: Economic and Policy Implications, WEIS 2007 - Sixth Workshop on Economics of Information Security, Pittsburgh PA, 7-8 June 2007.

[4] Ashish Arora and Christopher M. Forman and Anand Nandkumar and Rahul Telang, "Competitive and

Strategic Effects in the Timing of Patch Release", Fifth Workshop on the Economics of Information Security, 2006, Cambridge, UK

[5] Ashish Arora and Ramayya Krishnan and Anand Nandkumar and Rahul Telang and Yubao Yang, "Impact of Vulnerability Disclosure and Patch Availability – An Empirical Analysis", Third Workshop on the Economics of Information Security, 2004, Minneapolis, MN

[6] Karthik Kannan and Rahul Telang, "An Economic Analysis of Market for Software Vulnerabilities", Third Workshop on the Economics of Information Security, 2004, Minneapolis, MN

[7] Arbaugh, W.A; Fithen, W.L.; McHugh, John, "Windows of vulnerability: a case study analysis," Computer , vol.33, no.12, pp.52,59, Dec 2000

[8] Charles Miller, The legitimate vulnerability market: the secretive world of 0-day exploit sales, WEIS 2007 - Sixth Workshop on Economics of Information Security, Pittsburgh PA, 7-8 June 2007

[9] Michael Sutton and Frank Nagle, "Emerging Economic Models for Vulnerability Research", Fifth Workshop on the Economics of Information Security, 2006, Cambridge, UK

[10] Neil Gandal, Internet Security, Vulnerability Disclosure, & Software Provision, DIMACS Workshop on Information Security Economics January 18 - 19, 2007 DIMACS Center, Rutgers, NJ

[11] Jay Pil Choi and Chaim Fershtman and Neil Gandal, "Internet Security, Vulnerability Disclosure, and Software Provision", Fourth Workshop on the Economics of Information Security, 2005, Cambridge, MA

[12] Rahul Telang, and Sunil Wattal, "Impact of Software Vulnerability Announcements on the Market Value of Software Vendors – an Empirical Investigation", Fourth Workshop on the Economics of Information Security, 2005, Cambridge, MA

[13] Andy Ozment, "The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting", Fourth Workshop on the Economics of Information Security, 2005, Cambridge, MA

[14] Ashish Arora and Ramayya Krishnan Rahul Telang and Yubao Yang, "An Empirical Analysis of Vendor Response to Disclosure Policy", Fourth Workshop on the Economics of Information Security, 2005, Cambridge, MA

[15] Ashish Arora and Rahul Telang and Hao Xu, "Optimal Policy for Software Vulnerability Disclosure", Third Workshop on the Economics of Information Security, 2004, Minneapolis, MN

[16] SEC CF Disclosure Guidance: Topic Number 2: Cybersecurity (Oct. 13, 2011)

[17] http://www.cert.org/vulnerability-analysis/vul-disclosure.cfm (retrieved November 25, 2014)

[18] Interview with Eugene Spafford, conducted December 2, 2014

[19] Interview with Thomas A. Longstaff, conducted December 5, 2014

[20] United States of America v. Elcom Ltd., also known as ElcomSoft Co. Ltd., and Dmitry Sklyarov

[21] 18 U.S.C.A. § 793. Gathering, transmitting, or losing defense information

[22] Hasan Cavusoglu, Huseyin Cavusoglu, and Srinivasan Raghunathan. 2007. Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge. IEEE Trans. Softw. Eng. 33, 3 (March 2007), 171-185. DOI=10.1109/TSE.2007.26 http://dx.doi.org/10.1109/TSE.2007.26

[23] Krebs, Brian, "The Scrap Value of a Hacked PC", KrebsOnSecurity 2012, http://krebsonsecurity.com/2012/10/the-scrap-value-of-a-hacked-pc-revisited/

[24] Greenberg, A., "Meet The Hackers Who Sell Spies The Tools To Crack Your PC (And Get Paid Six-Figure Fees)", Forbes 2012, http://www.forbes.com/sites/andygreenberg/2012/03/21/meet-the-hackers-who-sell-spies-the-tools-to-crack-your-pc-and-get-paid-six-figure-fees/

[25] (Article comparing Shellshock and Heartbleed payloads)

[26] Cencini, A., Yu, K., & Chan, T. (2005). Software Vulnerabilities: Full- , Responsible- , and Non-Disclosure

[27] http://www.tux.org/lkml/#s3 (Linux Kernel Mailing List FAQ, including well-known public archives) (retrieved December 12, 2014)

[28] http://oss-security.openwall.org/wiki/mailing-lists/distros

[29] Silver-Greenberg, J., Goldstein, M., Perlroth, N., "JPMorgan Chase Hacking Affects 76 Million Households", The New York Times, 2014, http://dealbook.nytimes.com/2014/10/02/jpmorgan-discovers-further-cyber-security-issues/?_r=0

[30] Pub.L. 111-5, Health Information Technology for Economic and Clinical Health (HITECH) Act, HIPAA Privacy Requirements

[31] California's Breach Notification Law (the first of many), Cal. Civ. Code 1798.82

[32] https://blog.cloudflare.com/inside-shellshock/ (retrieved December 12, 2014)

[33] http://blog.cloudflare.com/answering-the-critical-question-can-you-get-private-ssl-keys-using-heartbleed/ (retrieved December 12, 2014)

[34] http://www.theregister.co.uk/2014/10/03/shellshock_bored_hackers_giving_up_droves/ (retrieved December 12, 2014)

[35] https://cve.mitre.org/data/downloads/index.html (retrieved November 25, 2014)

[36] Greenberg, A., "Shopping For Zero-Days: A Price List For Hackers' Secret Software Exploits", Forbes 2012, http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/

[37] Keizer, G., "Researchers rake in $280K at Pwn2Own hacking contest", Computerworld 2013, http://www.computerworld.com/article/2496004/security0/researchers-rake-in–280k-at-pwn2own-hacking-contest.html

[38] Zero Day Initiative Disclosure Policy (http://www.zerodayinitiative.com/advisories/disclosure_policy/)

[39] L Jean Camp and Catherine Wolfram, “Pricing Security”, Proceedings of the CERT Information Survivability Workshop, 2000 Oct 24-26, pp. 31-39, Boston, MA, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=894966

[40] Dmitri Nizovtsev and Marie Thursby, Economic Analysis of Incentives to Disclose Software Vulnerabilities, Fourth Workshop on the Economics of Information Security, 2005, Cambridge, MA, available online, at http://infosecon.net/workshop/pdf/20.pdf

[41] Sam Ransbotham (Boston College) An Empirical Analysis of Exploitation Attempts based on Vulnerabilities in Open Source Software The Ninth Workshop on the Economics of Information Security (WEIS 2010) Harvard University, USA 7-8 June 2010

[42] An Exploratory Study of White Hat Behaviors in a Web Vulnerability Disclosure Program Proceeding SIW ’14 Proceedings of the 2014 ACM Workshop on Security Information Workers

[43] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage and Geoffrey M. Voelker, Beyond heuristics: learning to classify vulnerabilities and predict exploitsKerberos ’95 placeholder

[44] [Google offers infinity million dollars, http://abcnews.go.com/Technology/google-offering-hackers-infinity-million-dollars/story?id=29215483]

[45] [http://www.pcworld.com/article/2874378/googles-project-zero-publishes-three-os-x-zero-day-vulnerabilities.html]

[46] [https://code.google.com/p/google-security-research/issues/detail?id=130&can=1&q=OS%20X%20status%3DNew]

[47] Acquisti, Alessandro; Friedman, Allan; and Telang, Rahul, "Is There a Cost to Privacy Breaches? An Event Study" (2006). ICIS 2006 Proceedings. Paper 94. http://aisel.aisnet.org/icis2006/94

[48] http://thomas.loc.gov/cgi-bin/query/z?c114:S.754:

[49] Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S., "Experimental Security Analysis of a Modern Automobile," Security and Privacy (SP), 2010 IEEE Symposium on , vol., no., pp.447,462, 16-19 May 2010 doi: 10.1109/SP.2010.34 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5504804&isnumber=5504699

[50] FireEye Labs, “APT30 and the Mechanics of a Long-Running Cyber Expionage Operation”, April 2015