MONTHLY PATCH RELEASE SCHEDULES: DO THE BENEFITS OUTWEIGH THE RISKS?

Dominic White, Deloitte and Barry Irwin, Rhodes University

dominicwhite@deloitte.co.za, b.irwin@ru.ac.za

ABSTRACT
This paper provides a comprehensive discussion on patch schedules. This discussion occurs over two parts. The first analyses existing implementations of patch schedules with a focus on Microsoft's monthly patch schedule. The arguments for patch schedules, namely increased patch quality and better planning within organisations are analysed and the impact of the type of disclosure investigated. It is concluded that in the case of delayed disclosure, where the vulnerability researcher privately discloses the vulnerability to the vendor allowing a patch to accompany the public disclosure, patch schedules provide significant benefits. However, in the case of instantaneous disclosure, where a vulnerability is disclosed directly to the public, as in the case of 0days, implementing a patch schedule significantly increases the risk to organisations waiting for a vendor patch. Some vendors already allow for 'out of band' patches to be released, however the criteria for choosing when to release a patch 'out of band' in unclear and often subjective. Additionally, involving the community in rapidly prototyping and testing patches will provide intrinsic benefits.

The second part then builds on these findings to provide advice to vendors implementing patch schedules. First the type of disclosure is recommended as an objective and pertinent criteria for differentiating when a patch should be released per a schedule or as soon as possible. Next, effective mechanisms for implementing both types of patch release are discussed.

The paper concludes that while patch schedules can provide significant benefits, vendors can still make many improvements based on recent examples to significantly improve their patch release methodology.

Some of this work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, THRIP, and the National Research Foundation with additional financial assistance from the DAAD foundation hereby acknowledged. Some work was undertaken while in the employ of Deloitte and their contribution is acknowledged and appreciated.

KEY WORDS
Patch Management, Patch Schedules, Best Practise, Risk

MONTHLY PATCH RELEASE SCHEDULES: DO THE BENEFITS OUTWEIGH THE RISKS?

1 INTRODUCTION

Effective policies are not only the responsibility of the users of software (end-users), software vendors must have a clear understanding of how they manage the patches they release and the best way to release them. Historically vulnerability disclosure and responding to vulnerabilities has proved difficult to standardise, with a high level of confusion and antagonism between security researchers and vendors. To combat this and ensure meaningful and useful interaction between researchers and vendors several disclosure policies have been suggested; a resource dedicated to collecting publications related to disclosure lists a total of twenty two different disclosure policies published between 1999 and 2004 [1] by vendors, security researchers and third parties. This confusion makes it difficult for vendors to standardise on a release policy, and instead the responsibility for formulating an effective patch management policy is passed onto the end-user. As will be demonstrated in this paper, this is because the type of disclosure has an impact on the effectiveness of a patch release policy.

In an effort to ease the administrative burden of patching on end-users some vendors have decided to move to a predictable patch release schedule. The first vendor to announce such a move was Microsoft. Soon afterwards Oracle and Adobe announced they would also move to a predictable cycle. John Pescatore of Gartner believes predictable patch release schedules are on their way to becoming an industry standard [2]. However, simplifying a patch release cycle ignores the complexities that the full disclosure debate has introduced. In both Microsoft and Oracle's case, the reactions to the announcements were varied. Some security experts were for the move [3, 4], others against [5] and the majority were silent, the lack of consensus indicated a shortage of research and understanding as to the possible effects. Since then both Microsoft and Oracle have both come under heavy criticism, and received praise for their patch schedule implementations by security professionals commenting on the same events. Propagating this policy to other vendors without a thorough analysis and with little understanding of the effects would not be desirable.

Surface observations of the implemented schedule have revealed both successes and failures. This paper provides a detailed argumentative analysis of patch release schedules, and their effectiveness. By examining examples of how various types of disclosure affects the risks faced by end-users, recommendations on how patch schedules should be implemented and when they are effective, or not, are formulated. In addition, lessons learned from recent public security incidents are used to suggest additional improvements to the process. The resulting observations are used to describe a method for other vendors to implement such a cycle that will both minimise risk and help ease the burden of patching on administrators.

2 STATE OF THE ART

In the past vendors operated without an obvious patch release schedule. When a vendor was notified of a vulnerability either through delayed disclosure or otherwise, the general approach was to create a patch and distributed it as soon as possible[1]. The problem with the "release when ready" approach is that it requires end-users to continually monitor patch and vulnerability announcements. The average systems administrator has to check for new security patches, usually daily or weekly depending on the available resources. This creates a situation where, combined with worsening number of vulnerabilities and additional problems created by patches, many administrators, either due to a lack of resources or will, just weren't installing patches effectively. Eschelbeck [6, 7, 8] is the only researcher at the time of writing to have provided empirical data demonstrating the impact of patch schedules. In 2004 Eschelbeck's data shows that it took 21 days to patch half the vulnerable machines on the internet after a patch was release (i.e. at 21 days 50% of vulnerable machines are patched), and 62 days for internal systems. Internal systems are increasingly vulnerable, due to the increased multiplexing of protocols over fewer ports, and content

[1]Some vendors had a more nuanced approach, however, this is not currently relevant and is discussed later

control decisions moving from the firewall to the end-user. Thus, internal systems have become necessary to protect as you would external systems, and this window from patch release to patch deployment (62 days) allows ample time for intrusions. Several notable examples of this have been large scale worm attacks such as the Code Red, Nimda, Sadmind, SQL Slammer, Blaster, Sasser, Witty and Zotob worms, which all showed significant numbers of internal 'desktop' machines infections. To combat this two high profile vendors, first Microsoft [9] and then Oracle [10] and more recently Adobe [2] chose to move to a monthly patch release schedule. The caveat was that critical patches could be released out of schedule, similar to the internal policy of some organisations where critical patches are given an expedited install plan. Microsoft chose to release patches on the second Tuesday of each month (a monthly release), while initially Oracle chose to follow suit, then changed to a quarterly release cycle [11]. However, Oracle have come under heavy criticism with some patches being released containing flaws up to three years after the vulnerability was announced [12]. Adobe, while planning to implement a monthly schedule, had not done so at the time of writing. Oracle's response to published vulnerabilities and quality of released patches has been poor. Most recently, Gartner came out severely criticising Oracle's patch practices [13]. Thus, given the lack of alternatives Microsoft provides the best implementation of a patch release schedule and will be the focus of the examples used, however this discussion is intended to be relevant to any vendor implementing a patch release schedule. In particular, this discussion applies to both open source and proprietary vendors.

The next iteration of Eschelbeck's research [8] showed that the scheduling appears to have improved things somewhat. In 2005 it took 19 days (down from 21) to patch half of the vulnerable machines on the internet, and 48 days (down from 61) to do the same for internal machines. The improvement in patching speed is provided in table 1. However, the improvement in patching is likely due to many other factors such as the renewed hype around patching, better patch and vulnerability notification and better automated patching tools, and cannot all be credited to patch schedules, especially since many vendors do not implement schedules as yet. The specific impact of scheduled patches was measured by Eschelbeck as being installed 18% faster. Additional statistics from Microsoft [14] indicate that the number of people applying Microsoft patches has improved dramatically (sometimes as high as 400%) since the change to a patch schedule. At first glance, the release schedule appears to be vindicated and proved as successful, however this research hypothesises that there are other intrinsic flaws in a patch release cycle that cannot be discounted.

	2003	**2004**	**2005**
External System's Half-Life	30 days	21 days	19 days
Internal System's Half-Life	N/A	62 days	48 days

Table 1: Half-Life of Vulnerabilities

3 AN ANALYSIS OF PATCH SCHEDULES

This section provides an argumentative analysis of patch schedules. An analysis of the specific effects schedules have when vulnerabilities are disclosed differently is provided. Some background is necessary for the discussion, namely what arguments the instigators of patch schedules provide and some background on the types of disclosure.

Specifically a patch schedule provides a predictable routine describing how often and when patches are to be released, with a constant time between patch releases. This is supposed to provide two primary benefits:

- Higher Quality Patches
- Better Patch Deployment Planning by End-Users

These improvements are advanced by vendors in the various press releases and discussion on implementing schedules [9, 10, 2]. There are other indirect benefits sometimes cited, such as faster deployment and greater patch deployment. However, these are knock-on effects of the improvement in quality and planning listed above and are not solely influenced by quality and end-user planning alone. For example, more detailed advisories, advertised to a wider audience could also result in faster deployment due to more readily available information for decision making, and greater deployment due to a wider demographic being aware of the patches. Thus, the focus will only be on the direct benefits claimed by vendors. The analysis below discusses what trade-offs occur in gaining these benefits, and if such trade-offs are acceptable. Most importantly, these benefits will provide ample justification for a patch release schedule if and only if they;

1. Are actually achieved
2. They are not achieved at the cost of a large increase in risk
3. They cannot be achieved through better means.

3.1 The Disclosure Debate

Before a discussion can be had arguing for the differences created in a schedule by different types of disclosure can be had, some background on the types of disclosure and the disclosure debate is necessary.

There are two primary types of disclosure, delayed disclosure and instantaneous disclosure. Delayed disclosure is often referred to as 'responsible disclosure'. Unfortunately, this is an emotionally laden term which is not always accurate and will be avoided in this discussion. There has been much debate in the internet community about the socially optimal method of disclosure. The full disclosure movement of the late 90's argued that by providing as much detail about a security vulnerability, the information was brought into the open and provided administrators with information with which to make their own security decisions. The introduction of the BugTraq[2] and Full Disclosure[3] mailing lists were an important part of this, where previously vulnerabilities were discussed in private between security professionals, now the information was freely available [15]. Arora *et al. [16]* state that proponents of full disclosure argue that it "increases public awareness, makes as much information public as needed for users to protect themselves against attacks, puts pressure on the vendors to issue high quality patches quickly, and improves the quality of software over time." The problem with full disclosure is that without an effective defence for the vulnerability, usually in the form of a patch, the information is of more use to malicious entities than to users [17]. Thus the concept of delayed or responsible disclosure was introduced, where the information is first released privately to a vendor and then disclosed publicly when the vendor releases a patch [15]. However, many vendors adopted an attitude of 'shooting the messenger', where researchers who disclosed the vulnerability were publicly slammed [18] for reporting on vulnerabilities that existed in the product whether they were reported or not. Most recently, Michael Lynn had his presentation at the Black Hat 2005 conference literally torn from conference proceedings and threats of legal action from Cisco systems for elaborating on a previously disclosed memory corruption vulnerabilities [19]. At the same time, vendors would sometimes excessively delay the release of a patch *[16]*. This led to much antagonism between vendors and security researchers. As a result third party trusted disclosure intermediaries such as CERT/CC were used to intervene in vulnerability disclosures, providing reasonable deadlines for vendors and ensuring security researchers disclosed 'responsibly' [17]. This also resulted in several recommended disclosure policies, with the more famous being Rain Forest Puppy's [20], the Organisation for Internet Safety's [21], Russ Cooper's NTBugTraq [22] and CERT/CC's [23]. Several papers have been written discussing the pros and cons of non-disclosure, full disclosure, partial disclosure and 'socially planned' disclosure *[16, 17, 15, 24, 25, 26, 27]*. A discussion on the various types of disclosure is beyond the scope of this section; a simple summary is that the debate has fallen on the side of delayed disclosure. It is sufficient to understand that there are two types of vulnerability disclosure, one in which the public becomes aware of the vulnerability when a patch is released and the other where the public and the vendor become aware when the vulnerability is released.

[2]`http://www.securityfocus.com/archive/1`
[3]`http://lists.grok.org.uk/mailman/listinfo/full-disclosure`

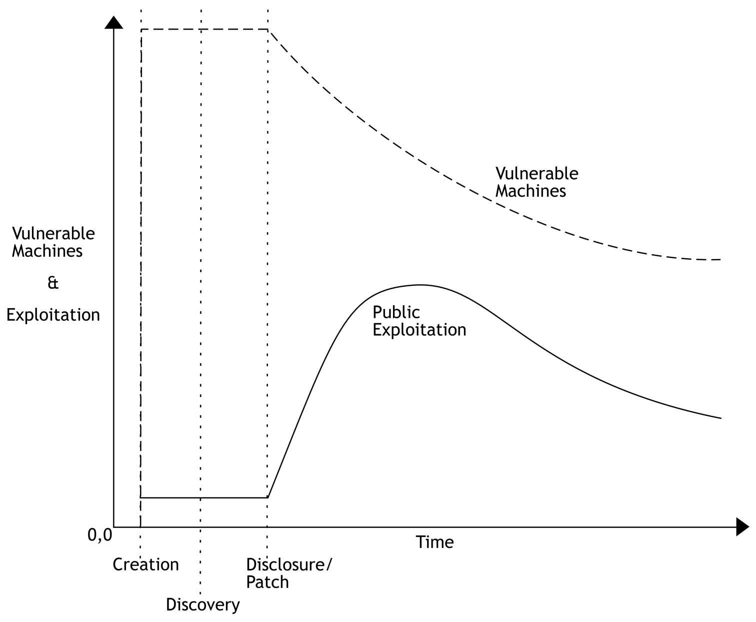

Figure 1: Delayed Disclosure and its effects on vulnerable machines and exploitation [26]

3.1.1 Delayed Disclosure

Figure 1 provides a visual depiction of a simplified vulnerability life-cycle in which the disclosure is delayed.[4]. The vulnerability is created when the software is first developed. At some point the vulnerability is discovered, this can happen multiple times and by different parties. The vulnerability is then privately reported to the relevant vendor and a patch is developed. At this time the only exploitation of the vulnerability occurs by the original discoverer and is of a limited scope. When the patch is ready, the vulnerability is publicly disclosed and corrected at the same time. At this point the number of vulnerable machines starts to decrease as patches are installed. At the same time the disclosure of the vulnerability details and the ease in which patches can be reverse engineered results in a rise in public exploitation of vulnerable machines. As the vulnerability and patch are publicised the number of vulnerable machines continues to decrease while the number of intrusions of still vulnerable machines continues to increase. A scripted exploit could be released soon after the release of the patch or longer. This will result cause a rise in the rate of exploitation, but is not relevant for the purposes of discussing the type of disclosure. It is sufficient to know that active exploitation is occurring, and is not included in the figure.

3.1.2 Instantaneous Disclosure

The process of instantaneous disclosure is similar to delayed disclosure, but with some pertinent differences. Figure 2 details the relevant events. Once again the vulnerability is created and at some point discovered. However, instead of reporting the vulnerability to the vendor the exploit is circulated within a community of black hats and private exploitation occurs. Sometime after this, the private exploitation is discovered 'in the wild' by a member of the public community and is reported to the vendor. At this point the process described in delayed disclosure occurs but with the difference that public and private exploitation occurs until a fix is released. The rate of exploitation will increase as the vulnerability is publicised and the exploit is possibly scripted, once again the increase in exploitation caused by the scripting of the exploit is not displayed. The number of vulnerable machines will only start to decrease once a patch has been released.

[4]The vulnerability life-cycle used here is simplified to highlight the differences between the types of disclosure, without muddying the waters with additional details.

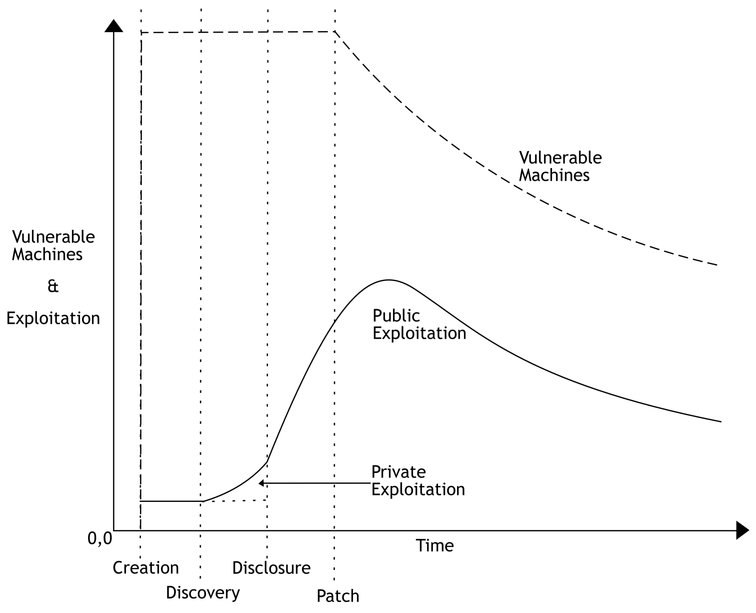

Figure 2: Instantaneous Disclosure and its effects on vulnerable machines and exploitation [26]

3.2 Patch Schedules and Delayed Disclosure

When the vendor has a choice as to when a vulnerability is publicly disclosed, the benefits of withholding the information until a patch is released are most obvious: The problem is acknowledged but a fix is available. It is important to remind the reader that open source projects also withhold vulnerability information from the general public until a patch can be developed. For example the Mozilla foundation frequently fixes 'Security-Sensitive' bugs which had not previously been disclosed [28]. A slight modification to ensure that these patches are released per a defined schedule brings more benefits. Administrators can avoid surprises and make plans ahead of time. Resources can be allocated, time scheduled and deployment planned. In addition the vendor can thoroughly test a patch to reduce the likelihood of a faulty patch being released without the pressures of attacks in the wild that need to be mitigated. With both the details of a vulnerability available and a patch which can be reverse engineered, a scripted exploited, whether released publicly or not, can be rapidly created [29]. This forces the vulnerability life-cycle to be synchronised with the patch release schedule. The only potential problem is that knowledge of the vulnerability may already exist within private and malicious groups or people[5]. This brings us to the original justification of full disclosure; by publicly announcing a vulnerability and encouraging people to patch, the number of attack vectors available to such groups are reduced. If there was no existing threat the vendor could silently fix the vulnerability in the next upgrade. The only defence from attacks against unknown vulnerabilities is a comprehensive defence in depth strategy which will hopefully mitigate or at least detect such an attack. Organisations currently face these threats, and releasing the patch per a schedule which results in the patch being delayed longer than if the vendor released it when ready, will not significantly increase the threat to an organisation from malicious attackers. This assumes there is limited exploit distribution within these 'underground' groups, a safe assumption in this case. Thus, the reduced threat from faulty patches and the increased efficiency of an organisation's patch management policy appear to more than justify this marginal increase in risk.

An important assumption is that the vendor develops the patch within a reasonable time frame. While the threats from an undisclosed vulnerability are limited, they are usually not zero. There is a potential for a separate discovery of the same vulnerability to occur by a malicious agent, or for the vulnerability to be 'leaked' by either the original researcher or agents within the vendor. The possibility of these events

[5]It is possible that the number of publicly disclosed vulnerabilities and the poor patching record of many organisations provides malicious groups with enough attack vectors without needing to research their own.

occurring increases over time and provides an incentive for a patch to be developed quickly. Thus, patch schedules with too long a wait between releases are likely to provide more than a marginal increase in risk and should be avoided. This is partly why Oracle is invalidated as providing a good implementation of a patch release cycle, as their quarterly release is too long. Unfortunately, there is little research into the probability of a leak occurring or a black hat discovering the same vulnerability, and this claim is based on an informed guess.

3.3 Patch Schedules and Instantaneous Disclosure

When vulnerabilities are disclosed irresponsibly the vendor no longer has control over when details of the vulnerability and a related exploit are released to the public. In the case of 0day exploits, a working exploit is made publicly available without providing the vendor with advanced warning. Similarly, if no proof of concept exploit was released with the vulnerability, the existence of a vulnerability for which there is no patch provides an attractive target and it can be assumed an exploit is not far off. Current research indicates that the release of a scripted exploit triggers the largest increase in attack activity [30]. Given the large increase in the threat level, minimising vulnerable organisations exposure is a priority for minimising risk. Thus, the critical factor becomes how soon the vulnerability can be effectively remediated. If a patch schedule will allow the patch to be released as soon as possible then it is vindicated. If however, the patch is delayed until the next release date instead of being released as soon as possible, this action is only justified if significant other benefits occur that cannot be achieved by any other means. The two benefits most commonly cited, as mentioned in the previous section, are that the delay due to the patch schedule allows more testing and allows administrators to plan for patch deployment. Both of these will be examined.

3.3.1 Quality

The argument for improved patch quality through more patch testing can be a persuasive one. The effort required by an organisation to minimise the risk of a patch causing problems are substantial and the single largest bottleneck of patch deployment[6]. Improving the quality of patches to a point where they could be deployed with little testing would substantially speed up patching and reduce risk. The argument is that by delaying the release of a patch, the vendor can engage in a thorough testing process. For example when a vulnerability in WMF files was discovered in the wild (a type of instantaneous disclosure exploit) [31], Microsoft's Security Response Centre had this to say about the patch:

> We have finished development of a security update to fix the vulnerability and are testing it to ensure quality and application compatibility. Our goal is to release the update [...] as part of the regular, monthly security update release cycle, although quality is the gating factor. [32]

However, the question must be asked: why must this testing be conducted in isolation? Surely collaboration with the wider community of end-users utilising the vendor's products would result in an increase in testing and wider test bed. For example, the benefits of community collaboration are well demonstrated by the activities of Lawrence Lessig, a Stanford professor of Law, who has been pioneering a movement named the *Creative Commons*. This movement seeks to encourage collaboration and remove the systems of control that seek to monopolise creativity. Lessig and his followers advocate a *remix culture* where the works of others can be freely used and built upon. One example of the benefits of such a culture were demonstrated when Lessig released his book *Free Culture* for free over the internet, something that until now would have been ludicrous to suggest to a publisher.

> Last year Penguin Press made an unprecedented move to release Lessig's 'Free Culture' under a [...] license that enabled people to freely download the book from the internet, and make derivatives for non-commercial purposes. After 24 hours, the book had been made available under 9 separate formats (txt, pdf etc.), after 36 hours, an audio version of the book

[6]This claim is based on the deduction that testing patches in all the necessary configurations present in the average organisation is usually far more effort than any of; reading notifications, making decisions, downloading the patch or deploying it with automated tool.

> had been announced, after 48 hours, a wiki had been launched [...] for others to build on and add to, and after one week, 200 000 copies of the book had been downloaded. Today, non-commercial translation projects have started in Chinese, Catalan, Danish, French, German, Italian, Polish, Portuguese (2) and Spanish (2). There are 3 audio versions of the book as well as versions for the Palm, MobiPocket and Newton. [33]

Since then several other derivatives have been created, including more ebook versions and several easy to use hyperlinked versions. However, such creativity and collaboration is not unique to publishing and is rather analogous to a recent event in the world of patching. When the WMF vulnerability for which no patch was available was discovered on the 27th of December 2005 [34, 35]; one day later initial anti-virus [36] and snort intrusion detection signatures [37] were available for the first variant; two days later a partial workaround for the vulnerability was posted [38], a movie of an exploit occurring was provided [38] and malicious sites exploiting the vulnerability were being shut down [39, 40]. Five days later a third-party patch was provided by Ilfak Guilfanov [41], later that day the patch had been disassembled and verified by the internet storm centre (ISC) which offered a digitally signed version [42]; a block-list of malicious sites and net-blocks utilising the exploit was created [43] and CERT provided a detailed vulnerability note on the issue [34]. Six days later a version of the unofficial patch was made available that allowed for an unattended install [44], it was distributed along with scripts for deploying the patch enterprise wide [45]. On the same day 'safe' versions of the exploit were provided for vulnerability testing [46] along with an executable vulnerability checker for vulnerability testing and patch verification [47]. The next day a comprehensive FAQ on the vulnerability was made available by the ISC [48], within a few hours this had been translated into 12 different languages, which had increased to 17 by the next day [49] along with presentations available in several different formats [48]. Eight days later the unofficial patch was made available as a Microsoft Installer Package (MSI) by Evan Anderson [50], for easier deployment, and this too was verified and signed by the ISC. Later that day the site hosting Guilfanov's patch experienced difficulty due to high load, a few hours later it had returned with 9 additional mirrors serving the files [51]. During this time, Microsoft maintained that an official patch would only be released on the 10th of January 2006 during the normal patch scheduled release [52]. After massive consumer pressure Microsoft eventually capitulated and released the patch on the 5th [53].

Why then did Microsoft not cooperate with this community in developing a patch? If knowledge of the vulnerability already existed then the benefits of keeping the patch confidential are lost, particularly when beta patches could be improved on and tested by such a wide and active community. Ironically, Microsoft possibly acknowledges this argument with their Security Update Validation Program (SUVP), which allows for patches to be beta tested within a chosen group of organisations, such as the US Air Force [54]. Microsoft benefits by the additional testing provided by an organisation with enough resources and an interest to thoroughly test patches, and in return the Air Force benefits from the early protection afforded by getting a jump start on their patch deployment process[7]. Although members of the SUVP are not allowed to use these beta patches in a production environment, they can benefit from early testing and ensuring their configuration is supported. There is no reason to assume these benefits would not scale if such a beta program was extended to include the public. A possible counter-argument to this is that a vendor can implement a better planned testing process, whereas testing within a community will involve a lot of redundancy and cannot be guaranteed to perform all necessary tests. However, this is simply a false dichotomy; all the benefits of a well planned vendor test schedule can be accrued in addition with testing input from a community. Tools and mechanisms allowing members of the community to interact and share their testing experiences already exists in the form of public mailing lists such as BugTraq[8] and PatchManagement[9]. The only modification required to take advantage of this testing community is to release the patches early and clearly mark them as unsupported beta's. By providing obvious warnings of the dangers inherent to deploying a beta patch, for example on the patch download site and in the actual patch's installer, or taking further steps such as providing a registration system, users who do not know better can be prevented from installing these beta patches.

[7]Given the ease with which exploits can be reverse engineered from patches, it is worrying to contemplate the American military being given such offensive capabilities before the rest of the world.

[8]`http://www.securityfocus.com/archive/1`

[9]`http://patchmanagement.org/`

The level of community involvement in response to the WMF vulnerability, particularly related to the unofficial patch, is unusual. While IDS and AV signatures and cooperation to shut down malicious sites are thankfully fairly standard, the community does not always get as involved as it did for the WMF vulnerability. The increased threat level of this vulnerability combined with confirmed inaction from Microsoft may have lead to the situation. However, while arguments claiming that one cannot always expect this level of community involvement are correct, this does not invalidate the point. If the community were to provide no additional help or guidance, an unlikely case, the vendor would still not lose anything by releasing beta patches and the community would at worse not benefit from the early release, but not lose anything either. If the vendor were to release details of which configurations the patch had been successfully tested on, the few who fulfilled those criteria could benefit from early patching without having to wait for all testing to be completed, ensuring that even if the community were of no help with testing, the exposure of some organisations could be minimised sooner.

3.3.2 Planned Deployment

> Having a predictable schedule makes it easier for customers to plan and when you can plan, it puts less stress on the customers' infrastructure and their people and the results are better. [55]

Providing a predictable patch release schedule can endear end-users to their vendor. The capability to plan and allocate resources ahead of time results in a much smoother deployment with less chance of errors. It moves patching from an emergency-mode procedure to an understood business process. Unfortunately, these benefits are once again only available if the vulnerability disclosure was delayed. Threat and vulnerability monitoring are a separate process from patch deployment. A patch schedule helps to synchronise the release of the patch, vulnerability and exploits so that threat, vulnerability and patch monitoring can likewise be synchronised. However, if the vulnerability was instantaneously disclosed the vendor is not able to maintain this synchronisation. Thus, an end-user needs to be constantly monitoring their network for attacks and understand and respond to potential threats. If a significant threat and vulnerability are discovered a risk assessment must be conducted and steps taken to mitigate the risk. This must be conducted whether the patch exists or not. Thus, the exact emergency mode scheduling patching seeks to avoid persists. The best way to "put less stress on the customers' infrastructure and people" is to provide an effective remediation as soon as possible. Placating end-users and playing down the threat to maintain the patch schedule instead of releasing a beta patch and encouraging community support to develop quality remedies is counter-intuitive. Even if the benefits of planning did apply in this situation, the corresponding increase in exposure is an unacceptable trade-off. This increase in exposure makes it more likely that an intrusion may occur. Intrusions are usually unscheduled and costly to recover from which would provide a greater inconvenience than deploying an unscheduled patch. The emphasis within the patch management community and this document is for an organisation to perform their own risk assessment and choose a course of action relevant to their needs. However, without the option of an effective remediation, a vendor would be severely limiting the organisations options for dealing with this risk.

3.3.3 Examples

The critical flaw in a patch release schedule is that it assumes all patches are responsibly disclosed. While the WMF vulnerability has provided the primary example used in the discussion above, there are other examples of instantaneously disclosed patches that have remained unpatched for a significant amount of time and resulted in a needless increase in an organisation's exposure to threats. Once again, the focus on Microsoft is unavoidable given the lack of any other vendor having effectively implemented a patch release cycle. The WMF vulnerability is unique in its level of community support and discussion, particularly from Microsoft who have been reluctant to discuss their motives in the past. Thus, the examples below are of vulnerabilities which could have been patched sooner, and were not for the sake of the patch schedule. However, they do not demonstrate the same level of community involvement as the WMF example above and contained no serious flaws, indicating that the testing within Microsoft is effective. Unfortunately, they do illustrate both the unacceptable increase in exposure and an inordinately large amount of time from vulnerability disclosure to patch release. It should be noted that these examples are illustrative of the failings of a patch schedule for instantaneously disclosed vulnerabilities only; Microsoft's patch schedule has proved quite effective for delayed disclosure vulnerabilities.

Krebs [56] researched the time it took Microsoft to release a patch from either the time of disclosure or the time it was reported to the vendor for 2003, 2004 and 2005. The dates and times were gathered by contacting the original researcher who discovered the vulnerability and Microsoft. Unfortunately Krebs calculations appear to be wrong [57] with inconsistent errors in the number of days from first disclosure until patch release and the number of patches counted. However, the dates he gathered appear correct, and once the calculations were fixed, because some days were too high and others low, his conclusions based on the averages remain true. The results appear in table 2, and show that when Microsoft moved to a scheduled deployment in 2004, the average time it took for a patch to be released for all vulnerabilities increased. They also show that for instantaneously disclosed vulnerabilities Microsoft has been getting faster at patching. Both these results make sense. The average time to produce a patch has increased due to the additional testing and quality assurance that occurs, and the average time to produce a patch for instantaneously disclosed vulnerabilities has decreased due to an increased security effort and an increase in threats. However, even at the lowest average of 46 days, this is far too long. This provides plenty of time for scripted exploits to be circulated and used by anyone including unskilled attackers. To reiterate, even if the patch quality is increased, the high exposure time brings this quality at too high a cost. By involving the community in the testing effort high quality patches can be produced sooner in this situation.

	2003	2004	2005
Number of Critical Patches	34	28	37
Average Days from Report to Patch	90.7	136	134
Average Days from Full Disclosure to Patch	73.6	55	46

Table 2: Microsoft Time to Patch Summary

Two examples of the type of damage that can occur during these long exposure times can be found in MS04-040 and MS05-054.

MS04-040 This Internet Explorer patch took 38 days to produce from the date of public disclosure. This vulnerability was not disclosed to the vendor before hand. The average time taken to release such a patch in 2004 was 55 days, thus, 38 days is well below the average. However during this time a variant of the MyDoom virus used the exploit as a propagation mechanism resulting in mass compromises. In addition, a banner-ad service was compromised and the exploit placed into the advertisements. These were then distributed across many high profile sites such as The Register and BBC leading to a substantial number of compromised machines [58]. As a final blow the Bofra/MyDoom mass mailing worm was developed and used the MS04-040 vulnerability to infect a machines [59]. These three large scale incidents occurred within these 38 days.

MS05-054 The original vulnerability related to this patch was publicly disclosed on May 28th 2005, however the vulnerability was described as a DoS attack and did not carry a high criticality. Microsoft still had not provided a patch after five months, at which point it was publicly disclosed, on November 21th, that the vulnerability could allow remote code execution raising its criticality. Proof of concept code was provided and soon afterwards the attack was detected in the wild. A patch to repair the vulnerability was only released on December 13th as part of the normal patch release. This means that the vendor had 177 days to develop a patch, but it still took 22 days to produce the patch once it had been discovered as critical.

3.4 Conclusion

The conclusion is quite simply that the arguments for a patch release schedule assume all vulnerability disclosure is delayed. The benefits claimed with a patch schedule are that a higher quality patch can be released and that end-users can better plan and schedule their deployments. However, when a vulnerability is disclosed instantaneously, these benefits are either lost, moot or could be better achieved. Patch quality could be achieved faster by utilising a community testing approach and scheduled patch deployments are not useful if it is likely to result in an unscheduled post-incident recovery.

4 ADVICE FOR IMPLEMENTING A PATCH RELEASE SCHEDULE

The prescribed policy is to have two release programs, one scheduled and predictable for delayed disclosure vulnerabilities and one immediate and collaborative for instantaneously disclosed vulnerabilities. This simple solution is similar to what is already supposedly implemented by vendors with their possibility of ’out of band’ patches. However, there are problems with the criteria used to differentiate between when a patch should be released per schedule or not. In addition, specific guidance as to how vendors can most help end-users and involve the community to increase patch quality faster is required. The policy discussed below provides a simple and effective method for releasing high quality patches and helping end-users minimise their risk. It first provides a clear criterion for discerning between which patch release mechanisms should be used. Then it details how each mechanism can be implemented, with reference to several current effective vendor practices.

4.1 Dual Schedules and Separation Criteria

As mentioned above, a vendor should utilise two release mechanisms. The first is a predictable and regular schedule with the other an unpredictable ’when ready’ release. One of the current criteria for distinguishing when to use which mechanisms appears to be risk. If a sufficiently large risk exists in the form of a significant threat then a patch will be released out of band. Threat is the deciding factor in the incomplete risk assessment conducted, as vulnerability appears to make little difference. When a worm is released or significant exploitation is detected, there is more pressure to release a patch out of band, often in the form of customer complaints and bad press reports. However, if an instantaneously disclosed vulnerability indicates that a significant portion of end-users will be vulnerable, then the pressure to patch only appears to come after a large threat is detected. For example, Microsoft’s defence of releasing the WMF patch as per scheduled indicated that their ’intelligence sources’ did not perceive a large threat, and only once significant customer pressure had been brought to bear was the patch released out of band. Thus, the current criteria can be extended to be one of either threat or external pressure. There are problems with these criteria. The problem with responding to threats is that a widespread and recognised threat does not negate the possibility or existence of targeted and specific attacks. Vendors should be seeking to minimise all vulnerability, not to minimise significant threats only. The problem with responding to external pressure is a similar one; once people are detecting attacks it is often too late, vendors should be seeking to prevent an attack in the first place. In addition, the size of the threat and external pressure are not an easy to measure and objective criterion. A vendor’s view of threats abstracted across all end-users is naturally a generalised one, so that while certain organisations may be facing significant threats and others none, the view to the vendor is only a medium threat. As for external pressure, the amount of ’noise’ one group makes is only tacitly linked to the actual problem. Thus a specific, objective, and measurable criterion is needed to differentiate between which release mechanism should be used. This document proposes that the form of disclosure be that criteria:

> If a vulnerability is disclosed responsibly then release the patch at the earliest possible scheduled release date. Alternatively, if a vulnerability has not been disclosed responsibly then release at the earliest possible date, ignoring the schedule.

This is the most relevant criteria if the arguments given above, which conclude that the benefits of patch scheduling only apply if a delayed disclosure is assumed, are taken into account. In addition, this criteria is trivially easy to determine and can be objectively judged by both the vendor and end-users. Vendors should adopt this as the discerning factor between a scheduled release and a critical release and clearly communicate this to their end-users to prevent misunderstandings.

4.2 Predictable Patch Release Schedule

Taking cognisance of the criteria above, the vendor should develop a regular schedule where patches for vulnerabilities which had their public disclosure delayed will be released. To reiterate, a delayed disclosure vulnerability is one which has been privately disclosed to the vendor. Most often researchers, who disclose vulnerabilities privately, will synchronise the release of their advisory for the time at which the vendor releases the patch. For example eEye security maintains a list of vulnerabilities [60] they have reported to vendors, for which a patch has not been released and they have been waiting to disclose their advisory. However, on occasion a researcher will specify a fixed date at which they will disclose their research. If negotiations fail and the fixed date is out of the schedule then the customers should be informed of the out of band release. This is a rare occurrence however, and is an example of why vendors should attempt to maintain good relationships with the security research community.

An important part of creating such a schedule is deciding on the length between patch releases. The difficulty in setting this length is twofold. The first is in choosing a length that reduces the time available for either the vulnerability to be discovered independently or leaked. The possibility of a vulnerability being discovered independently is only a concern for schedules that extends over several months. It is unlikely that such an extended schedule is necessary, as the majority of patches should not take long to develop and test, particularly since the critical release will require rapid patch development and testing. In addition, there is the possibility of delaying the release of a patch for a number of schedule iterations. For the same reasons that the schedule shouldn't have too long between iterations, there should be a maximum cap on the number of releases for which a patch can be delayed without very good reason. The second difficulty is in ensuring that the release cycle is optimised for all end-users. The deciding factor in this optimisation will be how often end-users can realistically afford to engage in patch management activities. Customer feedback and surveys should be conducted to gauge the optimal length. Bear in mind that customers will have a bias towards patching less often as it translates to less workload. This bias should be offset by the desire to minimise the potential of a leak or separate discovery, and to keep the number of patches deployed per release to a reasonable minimum, as offloading too many patches at once makes end-users risk assessments too complex, can impair the efficiency of monitoring efforts and exposes an organisation to too many threats at once. The current trend is towards a monthly patch cycle. A charitable assumption is that Microsoft, Oracle and Adobe engaged in comprehensive end-user discussion and the resulting choice of a month is optimised for the above values. However, the needs of customers, the frequency at which vulnerabilities are discovered and the speed at which patches can be developed are all dependant on the vendor, and as such this value cannot be generalised across all vendors.

One potential concern of an 'industry standard one month patch release' is that administrators may be flooded with several patches from separate vendors on the same day creating the same problems a vendor was trying to avoid. Alternatively, if the patches are released on different schedules at different times of the month, the problem of constantly applying patches which schedules try and minimise is re-created. This is a difficult problem that will affect end-users with multiple vendors for which vulnerabilities are regularly released. While automated patch deployment solutions will help with the deployment and installation of these patches, they provide little support for the larger and more time consuming problem of testing them. Ideally, end-users will standardise on manageable baselines. It will be in the vendor's interest to forge connections between vendors whose software is commonly used in conjunction with each other to ensure that the number of patches released at one time are kept to a minimum and interact correctly. In addition, planning for patches to be released within short gaps of each other would allow end-users to better plan deployment and manage threats than if all patches were released on the same day. While this 'multiple vendor' problem is quite limited at the moment, as more vulnerability research occurs and consequently the number of patches released grows, this problem may become worse in the future. Once such example of the multiple vendors problem was on July 12th 2005 when patches from Microsoft, Oracle, Mozilla and Apple were all released on the same day [61]. Granted, only two vendors engaged in a predictable release, but even if end-users had been aware off all the patches released, some end-users requiring all the patches would be forced into an awkward triage.

As privately disclosed vulnerabilities must remain private until a patch is available, a discreet, secure and confidential group of developers should be tasked with managing security patches and vulnerabilities. This is particularly true in open source vendors where the development is by its nature, open. The majority of vendors already have such a group implemented, and it is only mentioned here as a requirement in passing. The members of this group should be held accountable for any leaks and given the required access to ensure they can develop patches quickly. Given that patch development cannot be a task assigned to a small and constant group and by its nature spans all development and developers, mechanisms for temporarily bringing in other groups of developers, testers etc. need to be developed with the same levels of confidentiality and accountability.

4.3 Critical Patch Release

The critical patch release mechanism will seek to release a patch as soon as possible after the disclosure of an instantaneously disclosed vulnerability, where the vulnerability was not privately disclosed to the vendor before hand. In this situation the vendor would be informed of the vulnerability at the same time as the general public. This does not always occur through the release of a vulnerability advisory. A 0day exploit could be provided or a vulnerability advisory could be accompanied by proof-of-concept code. In all of these situations, a vulnerability has been instantaneously disclosed. Currently, some vendors already claim to have implemented such a critical release strategy. However, as discussed above, this release mechanism is only invoked at a subjective point determined by the vendor. In this version, the disclosure type of the vulnerability is the only appropriate discerning criteria. If a vulnerability has been privately disclosed and, before the chosen patch release date the vulnerability is either leaked or discovered independently and publicly disclosed, a decision to shift the patch from a scheduled release to a critical release should be made.

Once it has been determined that a patch should be fast-tracked and released as part of the critical patch release mechanism, a vendor should seek to engage the community of end-users to help ready a patch. The arguments discussed in section 3.3.1 described the benefits a community can provide, and how keeping the details of a patch secret until release are counter productive. The possible help a user community could provide is as limited as human imagination. Whether it is documentation, vulnerability scanners, workarounds, third party patches or vital testing; with the right motivation the skills of technical administrators can be leveraged. The work required in developing and delivering high quality patches has a high level of commonality across patches. This is not to say that the vulnerability and related fix are the same, but that all patches require, for example, testing and documentation. A vendor should enumerate the required tasks and highlight those where community support could provide a benefit. On-line collaboration tools to enable the community to engage in the required tasks should be provided. Most often these simply consist of an on-line forum; either a mailing list, forum software, wiki or bug tracking program such as bugzilla[10] can be employed. Peripheral benefits aside, the most specific and beneficial area of community involvement is in testing. By providing alpha or beta quality patches for early download, and sharing information on what has been successfully tested, a community can get involved. If multiple beta versions of a patch are to be released, enhancing or providing a patch roll-back mechanism would be one area where tools could be developed to aid testing.

A possible concern is that end-users would not be interested in deploying patches that are not at final release quality. However, end-users would not be applying beta patches directly to their systems. An effective patch management policy should always include a comprehensive testing strategy. In such a set-up no patch should be deployed without any testing, and the same would apply here. There are benefits to end-users getting involved in testing. By testing the patch on their specific configuration an end-user can ensure that the patch finally released works correctly for them. In addition, if a patch appears to function correctly it could be deployed early to machines that warrant it. Particularly since testing has a ’long tail’ where the initial work is in testing common configurations which apply to many users, whereas the later

[10] `http://bugzilla.org/`

tests usually only apply to a few users but require as much work. Once testing is completed on the common configurations, many users could deploy the patch sooner or at least get a head start on testing. For example, if a vulnerability primarily affects the Chinese version of a vendor's product, releasing the patch once the Chinese documentation is ready would allow the majority of users to start their deployment without having to wait for all translations of the documentation to be completed. The testing provided by the end-user community would allow the vendor to test different configurations faster, and the 'release-when-ready' approach would allow more end-users to deploy patches and hence decrease their vulnerability sooner. The only cost is a slight increase in the amount of testing performed by some end-users. However, the size of the community will usually help to ensure no one end-user's testing time increases dramatically, as the work is distributed and testing performed by one group can benefit many more with similar configurations. Thus, many end-users could continue as they do now and wait until the final release of the patch.

This release when ready approach can only help security by speeding the availability of vulnerability remedies. Only faulty patches being deployed on production machines would invalidate this. Thus, the vendor must emphasise that only the final production release of the patch should be deployed to production machines and all beta releases should be tested in a sand-boxed testing lab. There is then the possibility of two advantages. The first is that the testing feedback provided by the community will speed up the vendor's testing process resulting in a patch being available sooner. The second is that the patch, if it passed some configuration's testing, could be deployed sooner to some end-users without having to wait for every configuration to be tested.

The vendor should work hard to ensure all feedback is consolidated into a quality patch as soon as possible. With proper encouragement, embracing the community prototyping approach will help to cut down on the window of exposure from disclosure until a patch is available.

4.4 Encouraging Delayed Disclosure

Given the benefits evident when a patch release schedule is used for vulnerabilities which have had their disclosure delayed, it is in the vendor's interest to encourage delayed disclosure of vulnerabilities. Much discussion is available in each of the disclosure policies discussed earlier on how to maintain an amicable relationship between the vendor and security researcher. Vendors should make an effort to maintain positive relationships with the security community and vulnerability researchers in an effort to reduce the instances of instantaneous disclosure. Researchers too should consider how to best minimise risk to end-users when disclosing vulnerabilities, however that is outside the scope of this discussion. Two vendors contrast quite differently in their approach to this. Microsoft has done quite well in building its relationship with researchers over the last couple of years. There are few examples of recent public outcries by researchers who feel the vendor is not providing the patch within a reasonable time-frame. In addition, throwing parties for security researchers at conferences such as BlackHat [62] and outreach events such as BlueHat [63] have further helped to build a positive relationship. Oracle on the other hand has created controversy by taking too long to fix some bugs [64], and providing poor fixes even after these extended periods of time [65]. This has resulted in a negative perception of Oracle's patch release process and may decrease the chances of researchers working with the firm.

Another approach which has proved quite successful is the bug bounty program run by the Mozilla foundation [66], where $500 is awarded for each previously unknown security bug discovered in Mozilla software that is privately reported to the foundation. The foundation claims that the bounty program is working well. They have awarded $2 500 in bounties since its inception [66].

Additionally, relationships with security researchers can be smoothed by providing a clear and accessible description of how the vendor's organisation will respond when vulnerabilities are reported. Defining time frames in which contact will occur can help to manage the expectations of the researchers.

5 CONCLUSIONS

This paper has provided a discussion around the benefits and disadvantages of implementing a patch schedule. This discussion has provided *a priori* arguments on how patch schedules influence risk and are influenced by disclosure. These arguments have shown that patch schedules provide two benefits to end-users; the first is a higher quality patch with less chance of a fault, and the second is a predictable schedule which allows end-users to plan their resources and patch deployment reducing the surprise factor and helping to integrate patching as a normal business process. However, the argumentation also showed that these benefits do not accrue or come at too high a cost when the vulnerability has been instantaneously disclosed. The patch quality could be achieved better by releasing patches early as betas and gaining community support. Although, this cannot eliminate the ”surprise factor” in these instances due to the unpredictable nature of threats. To remedy this situation it is proposed that vendors maintain their patch schedule only for delayed disclosure. The type of disclosure forms a clear and objective differentiator for which patches should be scheduled and which shouldn't. In the past the differentiating factor had been a subjective threat assessment. In the situation of instantaneously disclosed vulnerabilities vendors should implement a critical release strategy that releases a beta of a patch to a community as soon as possible, allowing more testing to occur and providing benefits to end-users and the vendor.

References

[1] *Vulnerability disclosure publications and discussion tracking*. University of Oulu, Electrical and Information Engineering Department (May 10, 2005).
Available at: `http://www.ee.oulu.fi/research/ouspg/sage/disclosure-tracking/`

[2] McMillan, Robert. *Adobe Adopts Monthly Patch Cycle*. IDG News Service (December 15, 2005).
Available at: `http://www.pcworld.com/resource/article/0,aid,123935,pg,1,RSS,RSS,00.asp`

[3] Emigh, Jacqueline. *Users Weigh In on Oracle's Patch Plan*. eWeek.com News (August 23, 2004).
Available at: `http://www.eweek.com/article2/0,1895,1638797,00.asp`

[4] Livingston, Brian. *Microsoft's Patch-A-Month Club*. eWeek.com News (November 3, 2003).
Available at: `http://www.eweek.com/article2/0,1895,1490665,00.asp`

[5] Bott, Ed. *Patches: Once a month is not enough*. Ed Bott's Microsoft Report Blog (March 24, 2006).
Available at: `http://blogs.zdnet.com/Bott/?p=23`

[6] Eschelbeck, Gerhard. *The Laws of Vulnerabilities*. In *Black Hat Briefings* (edited by Jeff Moss). Black Hat, Inc, 2606 Second Avenue, 406, Seattle, WA 98121 USA (July 2003).

[7] Eschelbeck, Gerhard. *The Laws of Vulnerabilities*. In *Black Hat Briefings* (edited by Jeff Moss). Black Hat, Inc, 2606 Second Avenue, 406, Seattle, WA 98121 USA (March 2004).
Available at: `http://www.blackhat.com/presentations/bh-asia-04/bh-jp-04-pdfs/bh-jp-04-eschelbeck.pdf`

[8] Eschelbeck, Gerhard. *The Laws of Vulnerabilities 2005*. Qualys Research & Development (2005).
Available at: `http://www.qualys.com/research/rnd/vulnlaws/`

[9] Lemos, Robert. *Microsoft releases monthly security fixes*. CNET News.com (October 15, 2003).
Available at: `http://news.com.com/Microsoft+releases+monthly+security+fixes/2100-7355_3-5091835.html`

[10] Pruitt, Scarlet. *Oracle moves to monthly patching schedule*. IDG News Service (August 20, 2004).
Available at: `http://www.computerworld.com/securitytopics/security/story/0,10801,95388,00.html`

[11] Evers, Joris. *Oracle to deliver security patches on quarterly basis*. IDG News Service (November 18, 2004).
Available at: `http://www.infoworld.com/article/04/11/18/HNoraclepatchquarterly_1.html`

[12] Litchfield, David. *Opinion: Complete failure of Oracle security response and utter neglect of their responsibility to their customers*. BugTraq Mailing list (January 6, 2005).
Available at: `http://seclists.org/lists/bugtraq/2005/Oct/0056.html`

[13] Mogull, Rich. *Flaws Show Need to Update Oracle Product Management Practices*. *Technical report*, Gartner (January 23, 2006).
Available at: `http://www.gartner.com/DisplayDocument?ref=g_search&id=488567`

[14] Fisher, Dennis. *Changing Patch Habits With Microsoft*. eWeek.com News (December 6, 2004).
Available at: `http://www.eweek.com/article2/0,1895,1735542,00.asp`

[15] Farrow, Rik. *The Pros and Cons of Posting Vulnerabilities*. IT Architect (May 10, 2005).
Available at: `http://www.itarchitect.com/shared/article/showArticle.jhtml?articleId=8702916`

[16] Arora, Ashish; Telang, Rahul and Xu, Hao. *Optimal Policy for Software Vulnerability Disclosure*. In *Workshop on Economics and Information Security* (May 2004).
Available at: `http://www.heinz.cmu.edu/~rtelang/disclosure_finalMS_IS.pdf`

[17] Arora, Ashish; Krishnan, Ramayya; Nandkumar, Anand; Telang, Rahul and Yang, Yubao. *Impact of Vulnerability Disclosure and Patch Availability - An Empirical Analysis*. In *COMPLETE* (April 2004).
Available at: `http://www.heinz.cmu.edu/~rtelang/disclosure_finalMS_IS.pdf`

[18] Rauch, Jeremy. *The Future of Vulnerability Disclosure?* In *;login: the USENIX Association Newsletter*, volume 11 (December 8, 1999).
Available at: `http://www.usenix.org/publications/login/1999-11/features/disclosure.html`

[19] Schneier, Bruce. *Cisco Harasses Security Researcher*. CryptoGram Newsletter (July 29, 2005).
Available at: `http://www.schneier.com/blog/archives/2005/07/cisco_harasses.html`

[20] Puppy, Rain Forest. *Full Disclosure Policy (RFPolicy) v2.0*. Unofficial Policy (September 8, 2004).
Available at: `http://www.wiretrip.net/rfp/policy.html`

[21] *OIS Guidelines for Security Vulnerability Reporting and Response, V2.0*. *Technical report*, Organisation for Internet Safety (September 17, 2004).
Available at: `http://www.oisafety.org/guidelines/secresp.html`

[22] Cooper, Russ. *NTBugtraq Disclosure Policy*. *Technical report*, NTBugTraq (July 26, 1999).
Available at: `http://www.ntbugtraq.com/default.aspx?sid=1&pid=47&aid=48`

[23] *CERT/CC Vulnerability Disclosure Policy*. *Technical report*, CERT/CC (October 9, 2000).
Available at: `http://www.cert.org/kb/vul_disclosure.html`

[24] Laakso, Marko; Takanen, Ari and Roning, Juha. *Introducing constructive vulnerability disclosures*. In (COMPLETE COMPLETE).
Available at: `COMPLETE`

[25] Arora, Ashish; Krishnan, Ramayya; Telang, Rahul and Yang, Yubao. *An Empirical Analysis of Vendor Response to Disclosure Policy*. In *COMPLETE* (March 2004).
Available at: `COMPLETE`

[26] Rescorla, Eric. *Is Finding Security Holes a Good Idea?* In *IEEE Security and Privacy*, volume 3, no. 1: pages 14–19 (2005). ISSN 1540-7993. doi:http://dx.doi.org/10.1109/MSP.2005.17.

[27] Rescorla, Eric. *Security holes... Who cares?* In *Proceedings of the 12th USENIX Security Symposium*, pages 75–90 (August 2003).
Available at: `http://www.rtfm.com/upgrade.pdf`

[28] *Handling Mozilla Security Bugs*. *Technical report*, Mozilla Foundation (February 11, 2003).
Available at: `http://www.mozilla.org/projects/security/security-bugs-policy.html`

[29] Flake, Halvar. *SABRE BinDiff*. Product Website (June 26, 2005).
Available at: `http://www.sabre-security.com/products/bindiff.html`

[30] Arbaugh, William A.; Fithen, William L. and McHugh, John. *Windows of Vulnerability: A Case Study Analysis*. In *Computer*, volume 33, no. 12: pages 52–59 (2000). ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.889093.

[31] *Vulnerability in Graphics Rendering Engine Could Allow Remote Code Execution (912919)*. Microsoft Security Bulletin (January 5, 2006).
Available at: `http://www.microsoft.com/technet/security/bulletin/ms06-001.mspx`

[32] Kean, Kevin. *Updated Advisory: WMF Vulnerability*. Microsoft Security Response Centre (January 2006).
Available at: `COMPLETE`

[33] Ford, Heather. *COMPLETE*. Creative Commons South Africa News (COMPLETE COMPLETE).
Available at: `http://za.creativecommons.org/COMPLETE`

[34] Havrilla, Jeffrey S. and Dormann, Will. *Vulnerability Note VU#181038 Microsoft Windows Metafile handler SETABORTPROC GDI Escape vulnerability*. *Technical report*, US-CERT (January 20, 2006).
Available at: `http://www.kb.cert.org/vuls/id/181038`

[35] Anonymous. *Is this a new exploit?* BugTraq Mailinglist (December 27, 2005).
Available at: `http://archives.neohapsis.com/archives/bugtraq/2005-12/0305.html`

[36] *Exploit-WMF*. McAfee Virus Information Library (January 5, 2006).
Available at: `http://vil.mcafeesecurity.com/vil/content/v_137760.htm`

[37] *Bleeding Snort Current Events WMF Exploit Signature*. Bleeding Snort Current Events CVS Signature Repository (February 7, 2006).
Available at: `http://www.bleedingsnort.com/cgi-bin/viewcvs.cgi/sigs/CURRENT_EVENTS/CURRENT_WMF_Exploit`

[38] Carboni, Chris. *Update on Windows WMF 0-day*. SANS Internet Storm Centre Handler's Diary (December 29, 2005).
Available at: `http://isc.sans.org/diary.php?storyid=975`

[39] Wesemann, Daniel. *The most hated IP address of 2005 ?* SANS Internet Storm Centre Handler's Diary (December 28, 2005).
Available at: `http://isc.sans.org/diary.php?storyid=974`

[40] Serino, Jim. *RE: [Full-disclosure] Someone wasted a nice bug on spyware...* BugTraq Mailinglist (December 28, 2005).
Available at: `http://archives.neohapsis.com/archives/bugtraq/2005-12/0320.html`

[41] Guilfanov, Ilfak. *Windows WMF Metafile Vulnerability HotFix*. Hex Blog (December 31, 2005).
Available at: `http://www.hexblog.com/2005/12/wmf_vuln.html`

[42] Frantzen, Swa. *New exploit released for the WMF vulnerability*. SANS Internet Storm Centre Handler's Diary (January 1, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=992`

[43] Ullrich, Johannes. *Recommended Block List*. SANS Internet Storm Centre Handler's Diary (January 2, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=997`

[44] Sachs, Marcus. *Installing a Patch Silently*. SANS Internet Storm Centre Handler's Diary (January 2, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=1004`

[45] Sachs, Marcus. *Scripting the Unofficial .wmf Patch*. SANS Internet Storm Centre Handler's Diary (January 2, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=1008`

[46] Sachs, Marcus. *Checking for .wmf vulnerabilities*. SANS Internet Storm Centre Handler's Diary (January 2, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=1006`

[47] Guilfanov, Ilfak. *WMF Vulnerability Checker*. Hex Blog (January 1, 2006).
Available at: `http://www.hexblog.com/2006/01/wmf_vulnerability_checker.html`

[48] Frantzen, Swa. *WMF FAQ*. SANS Internet Storm Centre Handler's Diary (January 7, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=994`

[49] Sachs, Marcus. *.wmf FAQ Translations*. SANS Internet Storm Centre Handler's Diary (January 3, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=1005`

[50] Liston, Tom. *Updated version of Ilfak Guilfanov's patch / ,msi file*. SANS Internet Storm Centre Handler's Diary (January 1, 2006).
Available at: `http://isc.sans.org/diary.php?storyid=999`

[51] Hyppönen, Mikko. *Hexblog.com overloaded*. F-Secure Anti-Virus Weblog (January 4, 2006).
Available at: `http://www.f-secure.com/weblog/archives/archive-012006.html#00000767`

[52] Reavey, Mike. *WMF Vulnerability Security Update*. Microsoft Security Response Centre Blog (January 4, 2006).
Available at: `http://blogs.technet.com/msrc/archive/2006/01/04/416847.aspx`

[53] Nash, Mike. *Mike Nash on the Security Update for the WMF Vulnerability*. Microsoft Security Response Centre Blog (January 5, 2006).
Available at: `http://blogs.technet.com/msrc/archive/2006/01/05/416980.aspx`

[54] Mook, Nate. *US Govt. to Test Windows Patches Early*. BetaNews (March 11, 2005).
Available at: `http://www.betanews.com/article/US_Govt_to_Test_Windows_Patches_Early/1110560071`

[55] Nash, Mike. *COMPLETE*. Microsoft (COMPLETE COMPLETE).
Available at: `COMPLETE`

[56] Krebs, Brian. *A Time to Patch*. Washington Post's Security Fix (January 11, 2006).
Available at: `http://blogs.washingtonpost.com/securityfix/2006/01/a_timeline_of_m.html`

[57] White, Dominic. *Microsoft Patch Speed Inconsistencies*. .tHE pRODUCT Weblog (January 13, 2006).
Available at: `http://singe.rucus.net/blog/archives/687-Microsoft-Patch-Speed-Inconsistencies.html`

[58] Haugsness, Kyle. *Bofra/IFrame Exploits on More Web Sites (updated); IFRAME vulnerability summary; Two more IE Exploits*. SANS Internet Storm Center Handler's Diary (November 20, 2004).
Available at: `http://isc.sans.org/diary.php?date=2004-11-20`

[59] *Update for Microsoft Internet Explorer HTML Elements Vulnerability*. *Technical report*, US-CERT (December 3, 2004).
Available at: `http://www.us-cert.gov/cas/techalerts/TA04-336A.html`

[60] *Upcoming Advisories*. eEye Digital Security (January 2006).
Available at: `http://www.eeye.com/html/research/upcoming/`

[61] de Beaupre, Adrien. *Handler's Diary*. SANS Internet Storm Centre Handler's Diary (July 12, 2005).
Available at: `http://isc.sans.org/diary.php?date=2005-07-12`

[62] Toulouse, Stephen. *Microsoft presenting at the Black Hat security conference in Las Vegas*. Microsoft Security Response Centre Blog (June 9, 2006).
Available at: `http://blogs.technet.com/msrc/archive/2006/06/09/434600.aspx`

[63] Microsoft. *Microsoft BlueHat Security Briefings*. TechNet Security (March 8, 2006).
Available at: `http://www.microsoft.com/technet/security/bluehat/sessions/default.mspx`

[64] Vaas, Lisa. *Oracle's Silence on Database Security Wearing Thin*. eWeek.com News (August 17, 2004).
Available at: `http://www.eweek.com/article2/0,1895,1637079,00.asp`

[65] Vaas, Lisa. *Security Firm: Oracle Opatch Leaves Firms Uncovered*. eWeek.com News (August 22, 2005).
Available at: `http://www.eweek.com/article2/0,1895,1850287,00.asp`

[66] *Mozilla Foundation Awards Bug Bounties*. Mozilla Foundation News (March 28, 2005).
Available at: `http://www.mozilla.org/press/mozilla-2005-03-28.html`