Optimizing Network Patching Policy Decisions

Yolanta Beres, Jonathan Griffin

Optimizing Network Patching Policy Decisions

Yolanta Beres, Jonathan Griffin

HP Labs, Bristol, UK

{yolanta.beres, jonathan.griffin} @hp.com

Abstract. Patch management of networks is essential to mitigate the risks from the exploitation of vulnerabilities through malware and other attacks, but by setting too rigorous a patching policy for network devices the IT security team can also create burdens for IT operations or disruptions to the business. Different patch deployment timelines could be adopted with the aim of reducing this operational cost, but care must be taken not to substantially increase the risk of emergency disruption from potential exploits and attacks. In this paper we explore how the IT security policy choices regarding patching timelines can be made in terms of economically-based decisions, in which the aim is to minimize the expected overall costs to the organization from patching-related activity. We introduce a simple cost function that takes into account costs incurred from disruption caused by planned patching and from expected disruption caused by emergency patching. To explore the outcomes under different patching policies we apply a systems modelling approach and Monte Carlo style simulations. The results from the simulations show disruptions caused for a range of patch deployment timelines. These results together with the cost function are then used to identify the optimal patching timelines under different threat environment conditions and taking into account the organization's risk tolerance.

1 Introduction

Security decisions often involve trade-offs: a security policy choice that optimizes time spent by the security team might create burdens (cost) for IT operations or the business. One of the main tasks faced by the security operations team is vulnerability and patch management. The reasoning behind applying patches to remove system vulnerabilities thereby reducing security risk is well understood. The longer the systems stay unpatched the bigger the risk that a vulnerability may be exploited by malicious attacks or fast spreading malware. However, applying patches usually has many undesirable implications, mainly in terms of business disruptions. These disruptions are particularly high when patching network devices such as routers and switches, and especially so in such highly network reliant environments, as cloud infrastructures. Any time a piece of network equipment is patched, there is a risk of something going wrong: if the patch fails, or results in unexpected interaction with other devices or current configurations, the disruptions caused can have significant impact on the business, which relies on the network infrastructure. For example, patching Cisco devices usually requires the replacement of the complete device operating system; this is very different from the patching of Windows servers. After patching, it could easily turn out that the existing configuration does not work with the new OS version, or a routing protocol might end up being broken.

When setting their patching strategies, many organizations adopt regular patch update cycles, where the patches are required to be deployed once within a set time period, be it within one, three, or six months. Since patching automation for network devices is currently still technically difficult[1], this time-bound schedule is used to plan and allocate time periods for manually patching all necessary network devices within the schedule. The chosen patching schedule has to optimally address two objectives: minimize business disruptions from planned patches and upgrades, and minimize the time and the number of devices that remain exposed due to being unpatched for a long time.

These two objectives present a conflict, however. To reduce the number of planned disruptions due to patching the operations staff would prefer to adopt patching schedules that span longer periods of time, as more time can be spent on thoroughly testing each patch and also due to the fact that several patches may be released by the vendor when waiting longer and so several patches can be batched together and applied at the same time, reducing the level

[1] Due to high variability of network device OS types, and due to the fact that many patches require upgrades of OS, and thus might require multiple reboots, and possible re-configurations.

of disruption from patching. However, from a threat perspective the longer policy would increase the exposure of network devices to potential exploits and attacks due to many devices remaining vulnerable for long periods.

In this paper we explore how IT security policy choices regarding patching schedules for network equipment can be made in terms of economically-based decisions, in which the aim is to minimize the expected overall cost to the organization from patching-related activity. We apply a methodology that combines methods from economics—specifically, cost and utility functions—together with simulations of the executable system model capturing the underlying processes.

First, we introduce a simple cost function that takes into account costs incurred from disruption caused by planned patching and from expected disruption by an emergency when an exploit or malware emerges. We then construct a system model of the patch management processes, which is executed in the context of a stochastic model of the threat environment. To gain insight into the actual network patch management processes, we have worked with security teams and network operations staff across a couple of large organizations. In our model, we capture the main attributes of this process, but include some simplifications to make the model computationally feasible. We also make assumptions about the characteristics of the threat environment specific to network exploits and attacks, mostly based on historical (though sparse) data on network exploits over the past several years since 2004.

Finally we perform the experimental simulations, and show how changes in the patch deployment schedules affect the planned and unplanned disruption levels. These results together with the cost function are then used to suggest the optimal patching schedules for different tolerance levels of disruption and risk. Since the threat environment is ever changing, we perform additional simulations to show how the patching schedules should be adjusted under worsening threat conditions.

Our paper is organized as follows. Section 2 describes the network patching problem and introduces the cost function. In section 3, we present the model, constructed to capture the patch management process in the network environment. Section 3 also describes how we model the external threat environment. In section 4, we describe results and analysis from a set of simulation experiments based on the constructed model and under the assumptions of the introduced cost function. The analysis shows the changing levels of planned patching and emergency disruptions under different patch deployment timelines, and suggests some optimal timelines for the certain emergency tolerance level. Section 5 describes results from another set of experiments with a changing threat environment, and looks at how this affects the optimal timeline choice. In section 6, we discuss the implications of our analysis and some future work. Section 7 reviews related work in this area. Our paper finishes with some final conclusions.

2 Disruption Trade-Offs

In this paper we examine two types of disruptions that security teams dealing with vulnerability management across network devices have to encounter:

1. Planned operational disruptions that are caused by the device downtime due to patching. These can range from relatively short downtime due to device reboot to longer downtime where the patch failed and requires re-application or changes in the system configuration.
2. Unplanned disruptions that are caused by deployment of emergency procedures whenever there is emergence of exploit or malware. Even larger scale disruptions are encountered when the network is actually hit by attack that exploits unpatched vulnerabilities. The cost of emergency procedures is much higher than the planned disruption caused by patching. These emergency procedures could encompass expedited patching, or deployment of emergency workarounds. And so the savings in reduced operational disruption achieved with longer patch deployment timelines have to be weighed against the potential increase these type of disruptions.

2.1 Cost Function

Based on the two forms of business disruption introduced above we define a cost function that is used to determine the acceptable trade-offs when choosing the patch deployment schedules.

We use the following notation:

- c_{patch} is the cost of applying a patch to a single device (or upgrading the OS of a device), which for the purpose of this paper is mainly the disruption caused to the business because the device is offline and not usable;

- $c_{emergency}$ is the cost of applying an expedited fix or a workaround to a single device in case of exploit or actual breach/attack; this again is mainly the disruption caused to the business because the device is not usable;
- $p_{emergency}(t)$ is the probability that an exploit will emerge during some time interval $[t_1,t_2]$, raising the need for an emergency. This probability varies at different points in time t, where t is the time elapsed from the vulnerability disclosure. In section 3.1 we will choose a specific probability density function for our model of the threat environment.

Since an individual organization has hundreds or thousands of devices that might be vulnerable to the same vulnerability and require patching, the overall cost of an individual task of patching is multiplied across the population of all vulnerable devices, denoted as $dev_{patched}$:

$$d_{patch} = c_{patch}\, dev_{patched}$$

We assume that the cost of applying a patch does not fluctuate significantly across different types of network equipment or from one patch to another. We recognize that in some cases this is a simplification as patch quality may vary, and some network devices have a more critical role and cause more disruption when offline, but to make our analysis and modeling generic rather than organization- or vendor-specific we assume that downtime during patch application is relatively similar across the vulnerable devices.

The cost of an emergency, however, is incurred only if a breach is imminent due to the emergence of an exploit or the detection of an actual attack, and so the emergency disruption is dependent on the number of vulnerable (upatched) devices at the time of emergency, denoted as $dev_{unpatched}(t)$. The meaning for the cost of the emergency that we use in this paper does not take into account disruptions caused by the actual attacks on the unpatched devices or the implications of more complex attacks that exploit vulnerabilities on the unpatched devices. For the purpose of our analysis we assume that the emergency disruption cost comes from emergency patching of network equipment which happens when an exploit is known about, but before an actual attack takes place (attacks are rare). The resulting disruption is similar in kind to that caused by planned/operational patching, but of much greater severity as the operations and security teams have to rush to deploy patches or emergency workarounds across the remaining upatched devices causing disruptions outside the agreed allowed downtime periods and more severely impacting organization's business processes.

The arrival of an emergency event is modeled by the probability $p_{emergency}(t)$. We define the disruption during emergencies, $d_{emergency}$, as the expected value of emergency disruption across the unpatched population of devices:

$$d_{emergency} = c_{emergency} \int dev_{unpatched}(t)\, p_{emergency}(t)\, dt$$

Combining the two types of disruption together, the expected overall disruption caused by vulnerability management through patching is the patching cost plus the expected cost of emergency:

$$D = d_{patch} + d_{emergency}$$

The cost of disruption from patching or in case of emergency is obviously organization- and patch-specific, but for our analysis we need to make some simplifications. We say that the disruption cost per device caused by emergency procedures is α times greater than the disruption caused by applying a patch. This allows us to state that:

$$c_{emergency} = \alpha\, c_{patch}$$

The actual value of α is organization-specific, and should be selected depending on organization's tolerance level for emergencies, including the procedures for the patch deployment escalations, deployment of workarounds, or redundancies built into the network that might minimize the disruptions caused by an actual attack.

By substituting this into previous equation we have:

$$D = c_{patch}\,(dev_{patched} + \alpha \int dev_{unpatched}(t)\, p_{emergency}(t)\, dt)$$

Since c_{patch} is constant, the overall disruption cost depends mainly on the number of devices requiring a patch and the expected number of devices that remain unpatched at the time of an emergency.

This cost is incurred for each patch or batch of patches released by the vendor, so if, for example, a vendor releases three patches in a year that apply across the same population of devices, the cost D triples. In one year an organization usually has to apply hundreds of patches across its various systems and applications. For network devices, the number of patches released by vendors is considerable smaller, usually in tens rather than in hundreds per year, which is still quite a large number, considering that a typical large organization might have hundreds or thousands of network devices.

2.2 Reducing the Cost of Disruption

The security team can reduce the cost of disruption by planning the patch deployment timelines and using patch batching capabilities to bundle several patches together and apply them in one shot. For example, by bundling two patches together the administrator would cause half as much disruption, as each device has to be touched once instead of twice. The number of patches that can be bundled together is dependent on the vendor's patch release lifecycle. In order to meet the deadlines set in the patch deployment policies the administrators would usually start applying one patch across the first set of devices, and once another patch is released will batch it with the previous patch and apply them together across the next set of devices.

Looking at our cost functions the aim of the batching of patches is to reduce the cost dpatch for each patch by reducing the number of devices that the patch has to be applied to individually. By incorporating the patch batching effect for each patch, we can subtract from the total patchable population of devices the population for which the patch can be batched with the next or a superseding patch:

$$d_{patch} = c_{patch}(dev_{patched} - dev_{batch\_patched})$$

We are making a simplification here by assuming that the disruption caused when applying the batch of patches is the same as when applying a single patch. If the batch includes many complex patches, this might not be exactly the case. However, it's common within the network context that the next set of network patches is actually a full upgrade that supersedes or includes any previous patches, and so the cost of applying a new upgrade is the same or is increased only by small delta.

By choosing appropriate patch deployment deadlines that correlate with vendor patch release schedules the aim of the security team would be to increase the size of population within $dev_{batch\_patched}$, and thus achieve lower overall patch disruption costs. If d_{patch} was the only cost within D the biggest reduction of cost would be waiting for as many patches of possible and applying them together.

The other cost within D, the cost of emergency disruption $d_{emergency}$, however, increases with longer patch deployment timelines, as the population of devices unpatched at the time of an emergency grows. The aim would be to identify the appropriate patch deployment deadline that decreases d_{patch} (the patch has to be applied across minimum number of devices) while not increasing considerably $d_{emergency}$ (minimum number of devices remaining unpatched in case of emergency), and thus minimizes the overall disruption D caused by patching.

3 A Systems Model of Network Vulnerability Management

To explore the effect of different patching timelines on the cost of disruption, we use systems modeling and simulations. With the systems modeling the aim is to accurately capture the characteristics and behavior of the vulnerability disclose-exploit-patch lifecycle [1, 12], including the patch testing and patch deployment stages. We use a systems modeling methodology based on process algebra and queuing theory [10] that has been developed for framing and exploring models of complex systems and processes. Within this approach the processes, such as patch deployment, are captured stochastically and events that cause changes in the process, such as vulnerability exploit or patch release, as discrete events whose occurrence is specified with probability distributions or as time-based cycles. The models themselves are developed using specially created Gnosis programming language [25] and are represented visually as an activity type diagrams, with each component encoding certain distinct features of the system.

3.1 Patch Release and Patch Management Processes

We have previously developed a model of the patch management processes to explore the risk exposure window across Wintel environment in an organization [3]. We have adapted this model for network environment by introducing new features such as slower patching rate through patch uptake function and allowing longer patch assessment and preparation time. We have also changed how the threat environment is expressed to reflect the different attacker and exploit developer behavior. Figure 1 shows the visual representation of the created model.

The model diagram should be interpreted as follows: (a) the star and circular shaped components represent the events that occur at regular intervals as defined with an event inter-arrival probability distribution; a circular component has in addition the parameter defining the probability of an event happening; (b) the rectangular shapes correspond to the process steps each with an internal logic, that consists of time duration or some manipulations of internal

parameters; (c) the rhombus shapes are if-type decision boxes, that return Boolean values true or false based on a parameter value; and (d) finally, the process dynamics of the model is captured by the arrows connecting events to process steps and decision boxes.

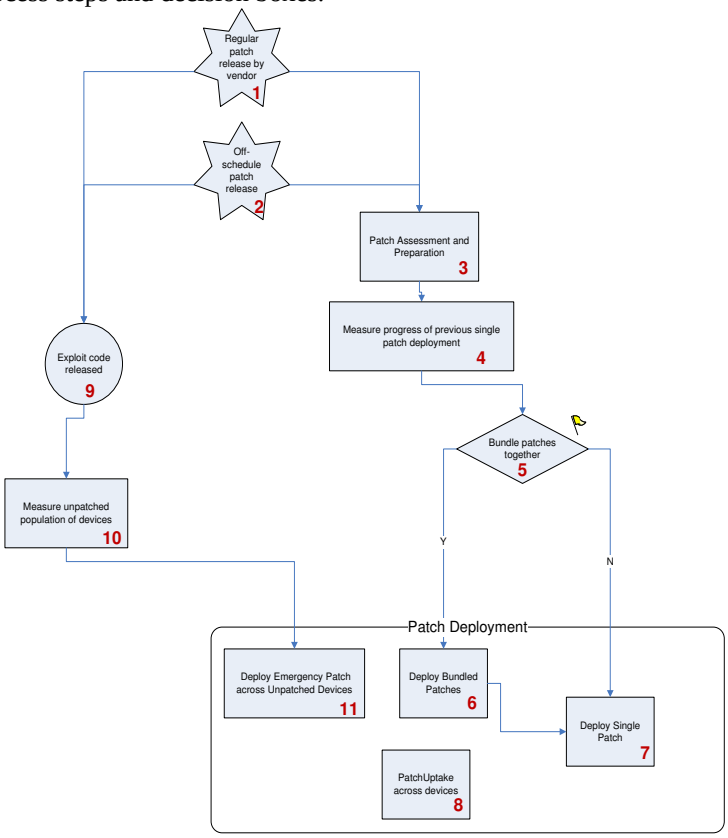

Fig. 1. Visual representation of the developed system model.

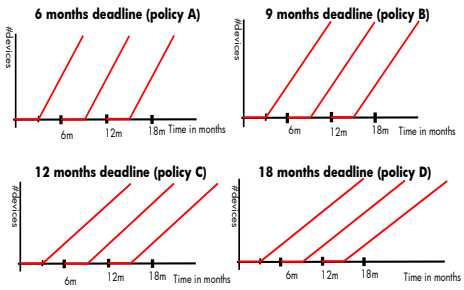

Fig. 2. Linear patch uptake across different patch deployment policies (deadlines).

The main trigger of a vulnerability management processes across networks is usually the release of a patch or a batch of patches by a vendor. This is different from Wintel type environments where such process is usually triggered by the vulnerabity announcements, since the threat environment and exploit discovery characteristics are not the same for network platforms. These are discussed in detail in section 3.2.

Some major network equipment vendors have recently adopted regular patch release cycles, and so we decided to include this in our model, but also recognize that off-cycle patches might be released in between. We have examined historical data in Secunia reports [15] regarding the patch release frequency by the three major network device vendors adopted across large enterprises: Cisco, HP ProCurve, and Juniper. Of these three, only Cisco has adopted a regular patch release policy with the main patches being released at 6 months intervals [4], in March and September, and some critical patches in between. Many fewer patches are released by HP ProCurve and Juniper, usually only 1 or 2 per year.

Since Cisco networking equipment is by far the most prevalent across large organizations, we decided to use the six-monthly cycle as the patch release frequency in the component (1) of our model: $f_{patch\_regular}(t_{patch}) = 60$.

We model the arrival of off-cycle patch releases as a Poisson process with an inter-arrival time based on an exponential probability distribution, with the mean time between occurrences set to one year. This is defined in the component (2) in the model: $f_{patch\_offcycle}(t_{patch}) = 365e^{-365t_{patch}}$.[2]

The process steps taken internally within an organization to manage patches consist of patch preparation and deployment stages. Based on interviews with the network administrators, we found that large organizations typically

[2] Everything is scaled in days in our model.

allow a fixed time for patch assessment and preparation; in our model this is set to be between 60 and 90 days. This parameter is defined for the component (3) as $f_{patch\_prepare}(t_{patch}) = U(60,90)$.

Once the patch has been assessed, we measure the progress of the previous patch (component (4)). If the patch has not been deployed across all vulnerable devices (decision component (5)), the patch is bundled with the previous patch and the bundle is deployed across that remaining population (component (6)). A single patch is deployed across all devices that already received the previous patch (if the previous patch has already been deployed across all devices, they would all receive a single patch until the next patch is released and assessed) (component (7)). The components (4)-(7) encode internal measurements, specifically recording the proportion of device population having bundled or single patches installed. The patch deployment is captured within the component (8).

For the patch deployment stage (8) we needed to determine if during a given time period for patch deployment across a number of devices, these devices are patched individually at regular intervals or in groups. By examining the patch deployment practices in several large organizations, we decided to generalize by assuming that in most cases the network devices would be patched individually at regular intervals so as to meet the deadlines set by the organization's policy. This would result in a linear patch uptake during patch deployment over the population of vulnerable systems, as shown in figure 2. We also assumed that the patch uptake has the same characteristics no matter what length the patching policy is set to; e.g. the devices would be patched at equally spaced intervals when the policy deadline is set at 6 months or at 18 months. Based on the lack of automated tools for patching network devices, and the limited number of allowed downtime periods set by the business, we consider that these two assumptions are not far off the actual network device patching practices.

3.2 The Threat Environment

We include the threat environment event in the model that cause an emergency when an exploit appears related to the vulnerability being patched. In choosing how to represent the threat environment in our model, we have examined previously announced cases of network exploits. As we have noted before, exploits on network devices are much less frequent compared to the Wintel environment due to the fact that network devices have many different CPU architectures and multiple ranges of platforms, thus preventing effective automatic exploit development. It is also much harder to reverse engineer the patches when hackers do not have access to the variety of router types or the necessary skills pool.

Up to now, the attacks and exploits that have been publicly announced have targeted specific versions of architecture and platform, making the likelihood of widespread attacks very low. Within the years 2002–2009 we have found several, though sketchy, publicly-announced instances of working exploits for Cisco vulnerabilities[3]. These exploits related to vulnerabilities for which a patch had been released over a year earlier by the vendor. Verizon report on attack vectors observed in 2008 [22] tells similar story, with only a small percentage of hacks observed that targeted routers, switches, or other network devices, and in cases when these exploited vulnerabilities, the patch has been available from a vendor a year or more before. This small number of working exploits together with anecdotal evidence from the hacker community [5, 6, 7, 9, 23] suggests that exploitation of network device vulnerabilities is generally difficult, with working exploits taking significant time to be developed after the patch publication by the vendor.

Based on this analysis, we made the assumption that exploits arrive relatively infrequently and some time after patch publication[4]. Therefore, when representing the threat environment within our system model, we concentrated on two factors, the rate of arrival of exploits, and how long after patch publication the exploit appears—which represents the time it takes for attackers/hackers to develop working exploit code. In the model, for each released patch we perform Bernoulli test on whether an exploit will be developed or not. The rate of arrival of exploits determines the likelihood that an exploit will be developed for any given patch, e.g., if the exploit arrival rate is one per year and the average number of patch releases is three per year, then the probability of an exploit being developed for each patch is 1/3. In the initial settings of the model we chose the exploit arrival rate as 1 emergency (significantly

[3] March 2004 toolkit exploited vulnerabilities from 2002-2003 [16], Nov 2006 SNMP exploit exploited vulnerability with patch available in 2004-2005 [17], Da Ios rootKit [24] in 2008, Yersinia toolkit released in 2005.

[4] We recognize that in some cases the vulnerabilities are announced and exploits could be available before the patch is released by the vendor. However, at the moment the evidence suggest that for network vulnerabilities these would be rare cases, especially for an effective exploit to be available much before patch release.

threatening exploit) every 2 years, as that seems to be closest to the anecdotal evidence available for Cisco vulnerabilities.

As described above, exploits for network equipment generally take some time to be developed. To capture this property we decided to use a Weibull distribution for the exploit development time, with a mean of 1 year and a standard deviation of 0.5 years (resulting in a shape parameter k=2.10 and a scale parameter l=1).

Thus component (9) in our model has two parameters, namely:

$$p_{emergency}(patch)=0.16 \text{ and } f_{exploit\_delay}(t_{exploit})=365 \text{ X } 2.10(t_{exploit})^{1.1}e^{-(t_{exploit})^{2.1}}$$

These basic settings for parameters regarding the prevailing threat environment are used in what we call the core simulation experiments, the results of which are described in section 4. To reflect potential changes in the threat environment such as increased exploit development rate, or in internal patching processes, we also make various changes in the parameters for additional experiments, which are described in section 5.

After exploit has been released, we measure in our model the proportion of unpatched population (component (10)) that needs emergency patch to be deployed (component (11)).

3.3 Measuring Disruption

During the simulations across different patch deployment schedules, we measure the overall disruption caused by normal patching and emergency procedures. We measure the total disruption caused per year, as this seemed a practical measure that can be used by the security teams in their policy decisions, since many security policies and budgets are determined on yearly basis.

As noted in section 3 the disruption caused by patching mainly depends on the size of the device population that requires a patch to be applied individually. During the simulations, we measure the relative proportion of the population rather than the exact number of devices, as comparisons across different patching policies were done based on the relative increase or decrease in disruption with different patching timelines, rather than the exact number. The same approach was applied for emergency disruption, where we record the proportion of the population remaining unpatched at the time of an exploit.

For example, throughout our set of experiments, the results of which are depicted across the charts in the next couple of sections, the disruption measured ranges from 0 to 10. The disruption of 1 means that full population of devices would be impacted by the operational or emergency patching. In such case, if an organization has 100 network devices, all of these 100 devices will be disrupted, usually by being offline for a certain period of time. The disruption of 3 means that full population of devices would be down three times per year. As was described in section 2, the disruption caused during emergency patching is assumed to be α times greater than the one caused during planned operational patching[5].

4 Results from Simulations with Core Model Settings

In the first set of simulation experiments, we look at how disruption changes with increasingly longer patch deployment timelines, with timelines increasing by one month at a time with a maximum of 24 months. The result of these simulations is plotted in figure 3. It shows the mean operational disruption from patching per year (d_{patch}), and the mean emergency disruption ($d_{emergency}$) as longer patch deployment timelines are being adopted by the security operations team.

As can be seen from these plots, with longer timelines the operational disruption decreases quite substantially, while the increase in the emergency disruption is more gradual and smaller. The savings are even bigger after the 6 month timeline. This point corresponds with the 6 month lifecycle when the vendor releases a new patch. For example, when the patch deployment time is set at 8 months the expected operational disruption is half that when patching policy requires patches to be applied immediately, corresponding to the timeline set at 0 months. However, for policies with timelines longer than 15 months the operational disruption improvements are smaller with each longer timeline.

As we recall from section 2, we made an assumption that the disruption cost per device caused by emergency procedures is α times greater than the disruption caused by applying a patch. We stated this as an equation:

[5] In the simplest case this would mean that during an emergency disruption a device is down for a longer period compared to the planned operational patch application, as the emergency workaround or patch is more likely to cause failures due to less effort spent on testing.

$c_{emergency} = \alpha\, c_{patch}$. When we scale the emergency disruption results by choosing the value α=10, we get the third line in the same graph. This is the case when an organization regards disruptions caused by the emergency procedures, be it the patch deployment escalations, deployment of workarounds, or the disruptions caused by an actual attack, as being ten times higher than disruptions caused by the planned patch application. In such cases within the graph we get a crossover point at a timeline of 9 months where $d_{patch} = d_{emergency}$. This represents the patching policy for which the overall disruption is caused by equal measures of operational patching disruption and emergency disruption.

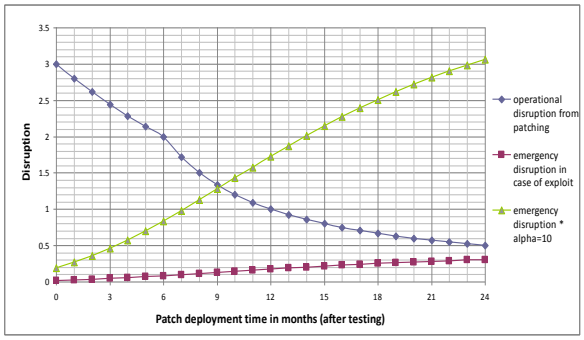

Fig. 3. Operational patching disruption and emergency disruption across changing deployment timeline.

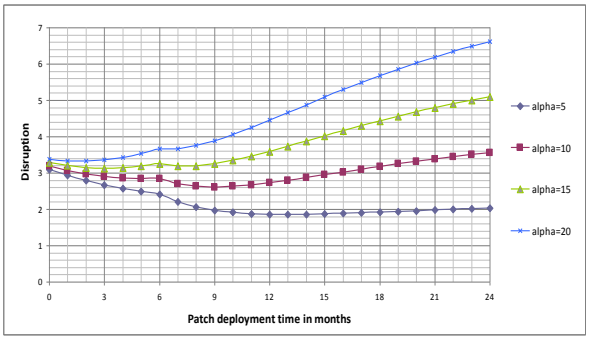

Fig. 2. Changes in the overall disruption (operational+emergency) under different values of α.

Based on our cost functions from section 2, the optimal patch deployment deadline would be the one where the expected cost of overall disruption, $D = d_{patch} + d_{emergency}$, is minimal for a certain value of α. As the value of α is dependent on a particular organization, its capabilities for dealing with an emergency (such as redundancies across network devices), and its risk appetite, we plotted the overall disruption D under different timelines for several values of α (figure 4).

When $\alpha=10$ the optimal policy is 9 months and this corresponds to the previous crossover point. If α is larger than that, the optimal policy deadline is much shorter, with $\alpha=15$ this being at 3 months and with $\alpha=20$ this being at 2 months. This means that for organizations where emergency disruption is regarded as being more than 10 times worse than the operational disruption caused by normal patch application, the policy should be adopted with patches required to be deployed within a month or two across vulnerable devices.

If, however, emergency disruption is regarded as similar to or just slightly worse than operational disruption6, the longer timelines would be more cost effective. For example, when α=5, the lowest overall disruption is achieved when the patch deployment deadline is set at 13 months. But even with timelines longer than that, the overall disruption increases only very slightly.

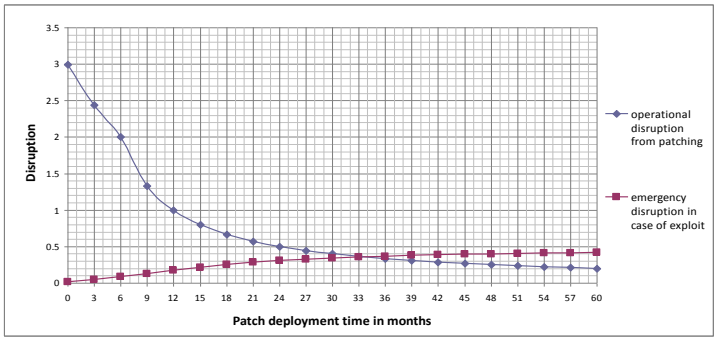

Fig. 4. Operational and emergency disruption with long patch deployment timelines

[6] This is quite likely the case within the current threat environment, where past exploits on network equipment have mostly resulted in denial of service (DoS) attacks, rather than attacks that give a complete access to the router or switch. The DoS attacks are usually not regarded as having big impact due to the redundancies that are commonly built in to the network architecture.

When we run experiments with even longer timelines, with maximum deployment time corresponding to 5 years, the results of which can be seen plotted in figure 5, a point is reached where the amount of emergency disruption exceeds the operational disruption. This is when the timelines are set to longer than 33 months. This suggests that with much longer patch deployment timelines the benefits of reduced disruption become smaller and smaller. The operational disruption cost is reduced only slightly over longer timelines while the emergency disruption cost does not increase, since, based on the assumptions in our model, after 2 years no more exploit is expected for a patch. Also at some point in time batching multiple patches together might not be feasible anymore as there might be physical restrictions that prevent successful installation of new upgrades on old hardware.

5 Changes in the Threat Environment

The results described in the previous section were generated from simulations in which the threat environment was as specified in section 4, corresponding to a mean exploit development time of 1 year after patch publication, and an arrival rate of exploits of one every 2 years. With some of the vendors of network equipment aiming to adopt more uniform OS architectures across their range of network devices, developing exploits that impact network devices may become much easier [8], and so the frequency of exploits may increase and the time taken for an exploit to be developed may decrease [13]. The policy which is optimal given current threat conditions may be far from best if the threat level changes. In the next set of simulation experiments we decided to explore how emergency disruption changes under a worsening threat environment, and how the policy deadlines should be adjusted so that to achieve minimal disruption costs.

5.1 Increased Arrival Rate of Exploits

First we increased the arrival rate of exploits, leaving the exploit development time the same as before set at 1 year. The changes in increased emergency disruption for various emergency arrival rates are plotted in figure 6. As can be seen from the chart, the emergency disruption increases considerably as the arrival rate increases, with the highest increase when an exploit appears every 6 months (0.5 year).

When we plot the overall disruption with α=10 in figure 7, we can see that with a worsening threat environment the previously optimal patch deployment policy of 9 months no longer applies. Although with the rate of arrival of exploits going up from one per 2 years to one per 1.5 years and one per year, we don't see a substantial increase in the overall disruption, with the rate at one per 0.5 years the increase is much bigger. The best option in such a case would be to patch immediately.

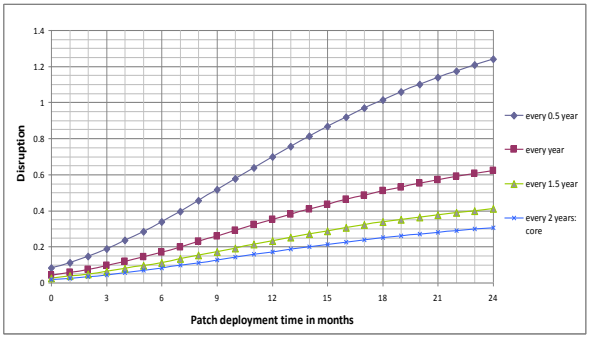

Fig. 6. Emergency disruption for different exploit arrival rates, with exploit development time constant at 1 year.

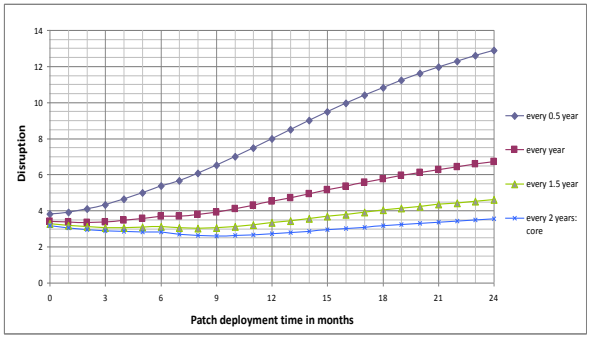

Fig. 5. Overall disruption when α=10 and exploit development time is 1 year for different exploit arrival rates.

5.2 Faster Exploit Development

When we reduce the mean exploit development time from the initial value of 1 year to 6 months or 3 months, the changes in overall disruption are quite significant, as seen in figure 8.

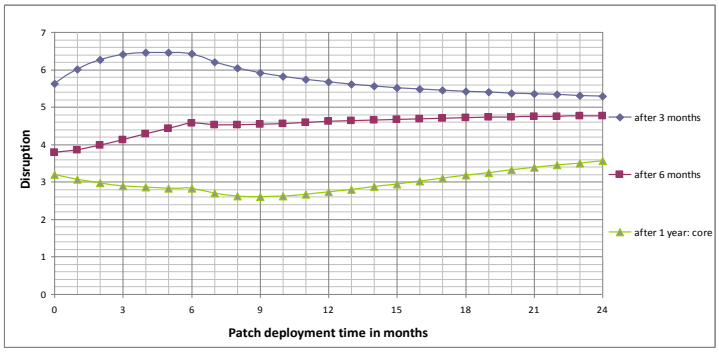

Fig. 7. Overall disruption when α=10 and emergency frequency is every 2 years, varying exploit development time.

The results for a mean development time of 6 months are as we might expect, with the earlier exploit arrival times increasing the expected level of emergency disruption, resulting in the optimal policy being to deploy patches as quickly as possible. The results for a mean development time of 3 months seem anomalous at first glance, until we observe that this is the same as the time taken for patch testing, so in this case a large proportion of emergencies arrive before patch deployment has even started. This limits the potential impact of changes in patch deployment timescales and under such threat environment conditions, both the vendors and the organizations would need to re-think the patch release and testing lifecycles and timelines or consider additional mitigation mechanisms.

6 Discussion

Our analysis explored the trade-off between operational and emergency disruption by recognizing that normal patching does introduce disruption, which is not negligible, and this disruption can be reduced with longer patching deadlines, that in turn would potentially increase the risk of a security emergency or breach. This analysis was done under certain assumptions about the parameters in the model. The results are most sensitive to the changing threat environment conditions, and that is why we explored the impact of these changes in detail in the previous section. As the threat environment gets worse, with exploits appearing frequently and soon after patch publication, the trade-off between operational and emergency disruption changes, until eventually the only option is to patch immediately. This would be the case for the Wintel environment, and the disruption from patching in such cases is small compared to emergency disruptions from the constant flow of malware and attacks. The reductions in overall disruption in such cases have to be achieved in a different way, maybe by implementing faster and less disruptive mitigation approaches. Therefore, the analysis in this paper is best suited to the context of vulnerability management for the network environment, or other types of environment where exploits and attacks are less frequent (e.g., some server environments, enterprise applications).

To help determine the appropriate timelines for patch deployment we chose to minimize the cost function that was defined in terms of disruption. In turn, we decided to simplify the definition of disruption so that it was mainly dependent on the size of the population of devices that would be impacted by the specific task: patching or emergency fix. Both of these can be predicted by running the simulations on the system model of the patch management process. To apply the cost function within the context of a specific organization the security team would need to choose only one parameter α, which is an estimation of how much worse the emergency fix is to the operations of the business than the planned patching.

Also our current interpretation of an emergency deals with an expected value that is only dependent on the number of devices unpatched and vulnerable at the time of the emergency. This might be too simple as the threats to networks become more sophisticated, and impact not only network devices, but critical business applications and transactions. So a more complex definition of emergency disruption might need to be developed, that takes into account the organization's risk appetite and ability to tolerate emergencies. This might also require a more complex system model that captures how an organization reacts to an emergency, including the effect of various processes and security mechanisms that are deployed to deal with the emergency.

7 Related Work

In recent years there has been some work examining the trade-offs involved in different patching policies, but none that specifically address the patching of network devices. Beattie, et al. [2] were the first to explore the factors affecting the best time to apply patches so that organizations minimize disruptions caused by defective patches. Their results indicate that patching during the period of 10 to 30 days after first patch release date is the optimal period for minimizing the disruption caused by defective patches. In our work, rather than looking just at a single patch and adjusting a single point in time to start patch deployment, we analyse the overall patch management process that also takes into account the time taken to apply the patches across all the vulnerable network devices in an organization; this can be considerable for a large organization.

Radianti, et al. [11] explore proactive and reactive patching policies using system dynamics modelling. Although their approach of modelling and simulation is similar to ours, the main difference is in the type of policies that are chosen to explore. Radianti, et al. explore generic patching policies, whereas we aimed specifically at exploring patching of network devices, as these represent a very special case.

Cavusoglu, et al. [18, 19] use a game-theoretic model to study interactions between a vendor releasing patches and an organization deploying the patches across its environment. They examine the cost/benefit consequences of the time-driven release policies adopted by a vendor and similar policies for patch deployment adopted by an organization, and explore situations in which the socially optimal patch management can be achieved. Under the condition that the vendor and patching organization each choose the policies that best suit themselves, they arrive at the conclusion that "vendors are better off releasing patches periodically" and similar regular patch update policies are the optimal strategies for the organization deploying patches. In our paper we assume from the beginning that both the vendor and the organization have adopted regular patch release and deployment cycles, as that has been observed as common practice among vendors and most of the organizations that we have worked with. However Cavusoglu, et al. assume that patch deployment takes an insignificant amount of time compared to the patch release period, whereas this is not the case for patching of network equipment where, as mentioned above, automated patching is not generally available, and patch deployment often takes significantly longer that the vendor's patch release period. Our analysis in this paper is specifically targeted at helping to identify the optimal time period for deploying patches as part of a regular patch cycle.

The work by Zhang, Tan and Dey [14] provides an analytical framework for cost-benefit analysis of different patching policies. They assume that patching lead time (the time taken to apply a patch across the systems) is negligible or very small comparing to the overall patching lifecycle, which we argue is not the case in large organizations with thousands of systems requiring the same patch. The authors also assume that the costs associated with vulnerability exploitations can be estimated with relative ease by an organization, which in reality is very difficult to determine with any accuracy. We believe that our proposed simulation-based approach allows an organization more naturally and flexibly to explore the pragmatic outcomes from different patching policies than a purely analytical framework would allow.

The topic on the cost of administration and cost of system downtime has received some attention [20, 21] and is aimed at identifying the cost to organizations of various administration and operation tasks. Patterson [20] suggests that the cost of downtime should be based upon calculation of two factors: revenue lost and work lost, but can become more complex and include such factors as morale and staff attrition. Couch, et al. [21] explore the actual administration cost incurred in response to various IT support requests. In our paper we take a simplified view of the cost of patching, this basically being the disruption caused to the business because the device is offline and not usable. However, for future work it might be useful to incorporate more complex definitions of cost in our analysis.

8 Conclusions

In this paper we explored how IT security policy choices regarding patching timelines for network equipment can be made in terms of economically-based decisions, in which the aim is to minimize the expected overall costs to the organization from patching-related activity.

We introduced a simple cost function that takes into account costs incurred from disruption caused by planned patching and from expected disruption by an emergency when an exploit or malware emerges. By lengthening the required patch deployment timelines, the IT security policy decision makers can reduce the disruption caused by planned patching as more patches can be batched together, but this would increase the expected emergency disrup-

tion as the devices would remain unpatched for longer. We applied a systems modelling and simulation approach to explore the disruptions caused by changing patch deployment timelines within the range of 0 to 24 months, and used the results together with the cost function to identify the optimal patching timeline. When modelling the network vulnerability management process we tried to capture current network patch management practices used across large organizations, and modeled the network threat environment based on historical (though sparse) data on network exploits over the past 4 years. The resulting optimal patch deployment policy of 9 months should be viewed as optimal under these assumptions. By increasing the frequency of exploits in the next set of experiments we saw that this 9 month timeline soon stops being an optimal policy, with the best option being to patch immediately.

We believe that the analysis described in this paper provides guidelines for the IT security policy decision makers in their respective organizations that can be applied when deciding on the network equipment patching policy that is optimal for their organization and their IT environment, and reflects their risk appetite and network emergency tolerance level. It is our hope that this approach may one day form best practice to follow not just in choosing patching policy but in other areas of security decision-making.

This work has been done as part of the Cloud Stewardship Economics Project [26], funded by Technology Strategy Board of UK Government.

9 References

1. W.A. Arbaugh, W.L. Fithem, J. McHugh, “Windows of Vulnerability: A Case Study Analysis”, IEEE Computer, 2000.
2. S. Beattie, S. Arnold, C. Cowan, P. Wagle, C. Wright, A.Shostack, “Timing the Application of Security Patches for Optimal Uptime”, *Proc of LISA’02: 16*th System Administration Conference, 2002.
3. Y. Beres, J. Griffin, M. Heitman, D. Markle, P. Ventura, “Analysing the Performance of Security Solutions to Reduce Vulnerability Exposure Windows”, Proc. of 2008 ACSAC, Dec 2008.
4. Cisco Security Advisories and Notices, “The publication schedule for Cisco Internetwork Operating System (IOS) Security Advisories”, March 2006.
5. Felix “FX” Lindner, “Cisco Vulnerabilities”, BlackHat Federal 2003.
6. Felix “FX” Lindner, “Development in Cisco IOS Forensics”, Defcon 11, 2003.
7. Technical interview, “Exploiting Cisco with FX”, 2005, http://www.securityfocus.com/columnists/351/2
8. Felix “FX” Lindner, “Cisco IOS-Attack and Defense: State of the Art”, 25th Chaos Communication Congress, December 2008.
9. M. Lynn, “The Holy Grail: Cisco IOS shellcode and exploitation techniques”, Black Hat USA, July 2005.
10. D. Pym, B. Monahan, “A Structural and Stochastic Modelling Philosophy for Systems Integrity”, HP Labs Technical Report, 2006.
11. J. Radianti, Finn Olav Sveen, J.J. Gonzalez, “Assessing Risks of Policies to Patch Software Vulnerabilities”, Proc. Of International System Dynamics Conference, July 2006.
12. B Schneier, “Managed Security Monitoring: Closing the Window of Exposure”, Counterpane.
13. Symantec Global Internet Security Threat Report: Trends for July–December 07, Volume XII, April 2008.
14. G. Zhang, Y. Tan, D. Dey, “Optimal Policies for Security Patch Management”, under review.
15. Secunia Advisories by Vendor, http://secunia.com/advisories/.
16. InfoWorld News, “Cisco warns of new hacking toolkit”, March 2004.
17. NetworkWorld News, “Can anyone stop the Cisco exploit?”, November 2006.
18. Cavusoglu, H., H. Cavusoglu, J. Zhang, “Economics of Security Patch Management”, Workshop on Economics of Information Security (WEIS), June 2006.
19. H. Cavusoglu, H. Cavusoglu, J. Zhang, "Security Patch Management—Share the Burden or Share the Damage?," Management Science, 54(1), April 2008.
20. D. Patterson, “A simple model of the cost of downtime”, In Proceedings of Large Installation System Administration Conference (LISA 2002), USENIX Assoc., 2002.
21. Alva L. Couch, Ning Wu, and Hengky Susanto, “Toward a Cost Model for System Administration”, In Proceedings of Large Installation System Administration Conference (LISA 2005), USENIX Assoc., 2005.
22. Verizon 2009 Data Breach Investigations Report.
23. Felix 'FX' Lindner, “Cisco IOS Router Exploitation”, Blackhat 2009.
24. S. 'topo' Muñiz, “Killing the myth of Cisco IOS rootkits”, May 2008, http://www.coresecurity.com/files/attachments/Killing_the_myth_of_Cisco_IOS_rootkits.pdf.
25. M. Collinson, B. Monahan, D. Pym, “A Discipline of Mathematical Systems Modelling”, HP and College Publications, 2012.
26. The Cloud Stewardship Economics Project, IISP, https://www.instisp.org/SSLPage.aspx?pid=463