Exploring vulnerability-induced attack frequencies through agent-based modelling

Yorick Paul Breukers

Faculty of Technology, Policy, and Management, Delft University of Technology
Cyber Advisory, Deloitte
y.p.breukers@student.tudelft.nl

Abstract. Software vulnerabilities are a major enabler for cyberattacks, and are therefore responsible for a sizeable portion of the risk for large organisations in cyberspace. Little research has been done on the dynamics underlying this risk. The goal of this paper is to find how organisations and software vendors can influence the overall risk level as a result of vulnerabilities. An agent-based model is developed to capture the behaviour and complexity of the vulnerability ecosystem. Somewhat counter intuitively, the model shows that not necessarily the number of vulnerabilities, but rather the diffusion of the knowledge on vulnerabilities, has a strong influence on the resulting attacks. Instead of trying to fix all vulnerabilities, focussing patching efforts on vulnerabilities that are actually being exploited can significantly reduce the risk of organisations. The model further showed that organisations should improve their own efforts to deploy patches on shorter notice to narrow down the window of vulnerability.

Keywords: vulnerability discovery, cyberattacks, risk exposure, agent-based modelling, patching

1 Introduction

Cyberspace is a major enabler for large organisations. However, the increasing reliance on digitalisation is not without risk. Securing the operations of large organisations in cyberspace is becoming increasingly challenging, as the number of incidents and the sophistication of the attacks increases. Software vulnerabilities are a major enabler for cyberattacks, and therefore account for a large proportion of the risk that organisations face. Understanding the relation between software vulnerabilities and the resulting cyberattacks should enable organisations to better deal with the risk of cyberattacks as a result of vulnerabilities.

Despite significant effort and progress made by the scientific community, surprisingly little is known on the impact of vulnerabilities and the resulting cyberattacks. More specifically, little is known on what kind of behaviour – from both the responsible software vendor and the organisations that use their software – has the most impact on the number of cyberattacks induced by software vulnerabilities. Understanding which behaviour has the most influence on the risk caused by vulnerabilities should help large organisations and software vendors to minimise that risk and improve their odds in cyberspace.

The goal of this paper is to explore which behaviour of software vendors and large organisations in response to discovered vulnerabilities, has the most influence on the number of cyberattacks. An agent-based model is developed to capture the complexity and behaviour of the system surrounding vulnerabilities. The model is used to analyse the influence of the properties that guide the behaviour of software vendors and organisations, on the attack frequency. This paper focusses on operating systems, due to available data and their importance for organisations in general.

The structure of this paper is as follows: Related work on vulnerability discovery and the resulting risk is discussed in section 2; the conceptualisation of the vulnerability ecosystem is presented in section 3; section 4 provides a brief description on the model implementation; the results from the simulations are presented in section 5, while their validity and implications are discussed in sections 6.

2 Related work

Despite the extensive research conducted on the dynamics behind vulnerabilities, few research has focussed on the risk induced by vulnerabilities. Some have focussed on modelling the vulnerability discovery process [6, 3, 5]. The goal of these methods is to determine the rate at which vulnerabilities are detected relative to the age of the software. They can be used to assess the risk associated with certain systems, due to the rate at which new vulnerabilities are likely to appear [5]. This should enable organisations to determine with some degree of certainty, how many vulnerabilities are likely to pose a threat in the near future. However, these models assume static code and can only be used for a single version of the software [33]. Another limitation is their inability to capture factors such as trend, level, and seasonality [35]. In general, these models rely on data directly related to the technical characteristics of the software and do not incorporate the broader context surrounding the software. Furthermore, they do not assess how the discovery of vulnerabilities relates to the number of attacks.

Other models have been proposed. One of those models determines the progression of the number of attacks that occur through a vulnerability [14]. However, this model requires data on reported attacks for that specific vulnerability, and can therefore only be applied to that vulnerability. A more elaborate model presented in [20] focusses on the attacks as a result of discovered vulnerabilities and the patching by the users. An important limitation of this model is that it omits the vulnerability black market. Other work takes a very detailed look at the discovery process and trade of vulnerabilities [2]. However, it does not take into account the complexity of patch release and patch deployment. A third model has been presented that focusses on the decision made by the discoverers on what to do with a discovered vulnerability [18]. It provides an extensive perspective on the potential options that the discoverer has. However, this model omits the important interactions between the vendors and organisations once a vulnerability becomes public, such as attacks, patch release, and patch deployment. Another model focusses on the time window between patch release and deployment [16]. However, given its game-theoretical nature, this model only focusses on a single vendor and organisation that uses its software. As a result, the

model misses the complex interactions between discoverers, exploits and mechanisms such as black markets and bounty programmes.

A final – though highly relevant – paper by [9], analyses the attack frequency as a result of the publication of a vulnerability through disclosure or patching. Based on empirical data, the paper argues that patching a vulnerability significantly increases the number of attacks. However, one of the limitations is that their analysis does not assess what kind of behaviour the software vendors and organisations exhibit, other than the publication and release of a patch by the vendor.

Despite valuable contributions, none of these studies cover all the elements of the vulnerability ecosystem. By taking a broader perspective on the system surrounding vulnerabilities, this paper analyses the influence of the behaviour of different actors on resulting cyberattacks.

3 The vulnerability ecosystem

The system surrounding vulnerabilities is obscure and complex. It is necessary to understand which actors and mechanisms influence the dynamics of vulnerability discovery, trade, and exploitation. This requires a conceptualisation that covers the most relevant concepts, in order to model the behaviour and analyse the consequences of that behaviour. A starting point for the conceptualisation is through the five phases in the lifecycle of a vulnerability: birth, discovery, disclosure, exploitation, and patch release [7]. Each phase is governed or initiated by different actors. These actors form the basis of the vulnerability ecosystem, since their behaviour determines what happens to the vulnerability during its lifecycle. An overview of the system is provided by figure 1; each actor in the vulnerability ecosystem has certain properties that guide its behaviour. Each actor, mechanism and object and their main properties are discussed in the remainder of this section.

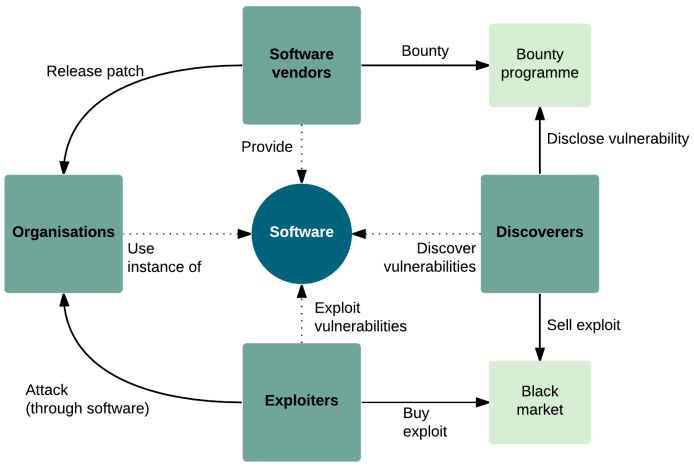

Fig. 1. Conceptualisation of the vulnerability ecosystem

Software vendors The software vendor has three basic actions that it performs in this system. First, it releases software with a certain quality. Second, it can develop a patch for a new vulnerability. Lastly, it releases the patch at a certain moment. These actions are guided by the properties of that specific vendor. The first property is the vendor's ability to reduce the number of vulnerabilities in the software. Developing completely secure software is difficult and prohibitively expensive. However, vendors can use methods like fuzzing to significantly reduce the number of vulnerabilities [26, 37]. Therefore, the vendor has to some extent, the capability to develop secure software, which reduces the number of vulnerabilities in the code when the software is released. The second property of the vendor determines how it responds to a new vulnerability; its patch development policy. This is the main determinant for whether a patch will be developed or not [36]. The decision to develop a patch is guided by the severity of the vulnerability and its risk of being exploited [8, 38]. Depending on the patch development policy, the probability that a patch is developed and the development time, varies among different vendors. The final property of the vendor is its patch release policy. Software vendors like Microsoft release patches on a monthly basis. The main argument for periodical patch releases is to allow organisations to plan the patch implementation and decrease the burden on system administrators [16]. In general, three types of release policies can be distinguished: periodical, semi-periodical, and continuous.

Organisations Within this system, deploying released patches is the primary action of the organisations that use the software. Numerous vulnerabilities are reported every day. Patching all those vulnerabilities has proven to be a challenge for large organisations, since it requires testing and deployment of the patch over countless systems within the organisation [10, 16]. The decision to deploy a patch is guided by the organisation's security strategy. Broadly speaking, we can distinguish between proactive versus reactive security investment strategies. Proactive refers to the practice of proactively analysing the threat landscape and investing accordingly. This method reduces security incidents, but it requires substantial upfront investment [25]. Organisations that adopt a reactive strategy wait and observe security incidents and use that knowledge to allocate security investments. This is more cost-efficient, and can therefore be preferred in some situations [12]. Additional properties are its value, which determines how attractive the organisation is to potential exploiters, and the number of systems within the organisation that use the software. The number of systems influences the time it takes for an organisation to deploy a patch [10].

Discoverers Due to the required skill, a clear distinction should be made between hackers that actively search for vulnerabilities (the discoverers) and hackers that actually exploit the vulnerabilities in order to attack a target (the exploiters) [2]. Vulnerability discovery is a highly skilled activity, which requires a high degree of technical skills and knowledge of the software [2, 26]. The chance of discovering a vulnerability is influenced by the skill level and knowledge level of the discoverer. By investing effort in discovering vulnerabilities, the knowledge level increases over time for that specific piece of software, and increases the odds of finding vulnerabilities [21]. The decision to disclose the vulnerability to the vendor, or to sell it on the black

market, is to a large extent guided by the motivation of the discoverer [18]. Although financial rewards remain a strong incentive for selling vulnerabilities [2], large price differences between the black market and legitimate alternatives [26] make any direct form of price-based competition unlikely. Therefore, the motivation of the discoverer primarily determines what happens to the vulnerability once it has been discovered.

Exploiters The final type of actor is the exploiter, which comprises a broad range of potential threats. The main action of the exploiter is to use exploits in order to attack an organisation. The exploiter's ability to attack an organisation depends on its access to exploits and its ability to actually use the exploit, which is influenced by whether or not the exploit is scripted [7]. Given the broad range of potential types of exploiters, it is assumed that there are two types of preferences for exploits. The first one is target-driven, which means that an exploiter has a preferred target and therefore requires an exploit for the software that is used by that target. The second type of exploit preference is availability-based. In this case, the exploiter targets software simply because the exploiter has access to related exploits. This can either be because they are included in automated tools like Kali-Lunix, or because they are easily available on the black market. In order to further simplify the broad range of potential exploiters, two levels of sophistication of the exploiter are assumed; advanced exploiters and basic exploiters. This determines their ability to use exploits, their ability to acquire exploits, and their chances of a successful attack.

Black markets and bounty programmes Although vendors take increasing measures to reduce vulnerabilities and increase their own capacity to discover vulnerabilities [26], the majority of the discovered vulnerabilities are found by discoverers external to the vendor [2]. Selling the knowledge on the presence of a vulnerability or the exploit, is not a straightforward process [29]. Vulnerability black markets facilitate the illicit sale of exploits and vulnerabilities. Obscurity, privacy, and reputation are important aspect of these marketplaces, which operate like a regular market, where supply and demand determine price [1]. Little is known about the inner workings of the black market; this includes the lack of knowledge on the pricing mechanisms within it. The knowledge asymmetry between the buyers and sellers on the black market increases the uncertainty for both parties [29], and adds to the challenge of defining a standardised pricing mechanisms.

It is suggested that the price of an exploit is determined by the severity of the vulnerability to which it applies, the amount of access it gives, and the supply and demand [26]. Due to the lack of knowledge on black market pricing mechanisms, we assume a pricing mechanism, based on the severity level of the vulnerability, the market share of the software, and the supply and demand.

The main alternative to the black markets are bounty programmes [2]. Bounty programmes become more popular and are economically more efficient than hiring full-time discoverers [17]. However, there are also problems with bounty programmes. Discoverers tend to move away from older bounty programmes in favour of newer ones, as they offer a higher chance of finding vulnerabilities [28]. This decreases the effectiveness of these programmes over the long run. Although there is increasing interest in bounty programmes, we assume that a bounty programme only transfers

the knowledge of a vulnerability from the discoverer to the appropriate vendor, thus omitting different types of bounty programmes.

Software and vulnerabilities Software plays a central role within the vulnerability ecosystem. Its first property is that it has a vendor which provides the software and patches. In addition, software is a certain version, which determines its relation to newer and older versions of the software. More important are the vulnerabilities that the software contains. This is influenced by the vendor's ability to develop secure software and reduce the number of vulnerabilities [26, 37]. The number of remaining vulnerabilities in the software has an influence on how easy it is to find new vulnerabilities, as this reduces the chance of finding new ones. However, constantly changing code, due to patches and upgrades, introduce new vulnerabilities [36]. Software thus has a number of vulnerabilities, which is decreased if a patch is implemented, while at the same time it can be increased by that same patch. An additional property of the software is the amount of shared code. Shared code between the older and newer version of the software accounts for a significant number of discovered vulnerabilities [22]. This means that the amount of shared code has an influence on the number of discovered vulnerabilities, since it improves the chance that a discovered vulnerability also applies to other versions.

A vulnerability is defined as *"a flaw or weakness in system security procedures, design, implementation, that could be exercised and result in a security breach or a violation of the system's security policy"* [32]. The main focus is on vulnerabilities that result from mistakes made by the software vendor. The first property that defines a vulnerability is the software to which it applies. The second property is the severity level of the vulnerability. This determines its potential value and potential risk to an organisation. Finally, a vulnerability has a certain status. Its status reflects the different states in the vulnerability lifecycle, as defined by [7].

4 The agent-based model

In order to determine what vendors and organisations should do to minimise the risk induced by vulnerabilities, it is necessary to explore the properties of the vendors and organisations which have the most influence on the attack frequency as a result of discovered vulnerabilities. Understanding this relation could be achieved through the analysis of empirical data. However, this requires data on attack frequencies, the properties of the involved vendors, the targeted organisations, and which vulnerabilities were exploited during the attacks. To the author's knowledge, no such dataset exists or is publicly available. For this reason, a modelling and simulation approach has been chosen to simulate the behaviour and interactions between vendors, organisations, discoverers, and exploiters.

The previous section described the system surrounding vulnerability discovery and the resulting attacks. It shows the complexity and how the behaviour of each actor influences the lifecycle of a vulnerability. Modelling such a system requires an approach that is able to capture individual actors with different properties and their behaviour as a result of a constantly changing environment. Agent-based modelling

is a suitable modelling method, due to its ability to capture emergent behaviour, enabling a natural description of the system, and its high flexibility [13]. In addition, agent-based modelling is recommended for situations where it is important that agents learn, adapt, behave strategically, and anticipate [34]. These characteristics make it a suitable method to model complex adaptive systems such as the vulnerability ecosystem. Despite its potential value, there are only a few examples of the application of agent-based modelling within the field of cybersecurity research [19, 23, 24]. None of these examples aim at understanding the complexity behind vulnerability discovery and the resulting cyberattacks.

4.1 Model implementation

The agent-based model is implemented using NetLogo [39]. The agents, as described in the conceptualisation, interact according to the sequence in figure 2. The flow chart provides a high-level overview of the procedures in the agent-based model. The procedures are ordered in such a way that a vulnerability goes through the different phases of its lifecycle in a natural order [7]. Each of these procedures covers different processes within the vulnerability ecosystem. For this paper however, four of these procedures are of particular interest. They concern the development, release, and deployment of a patch.

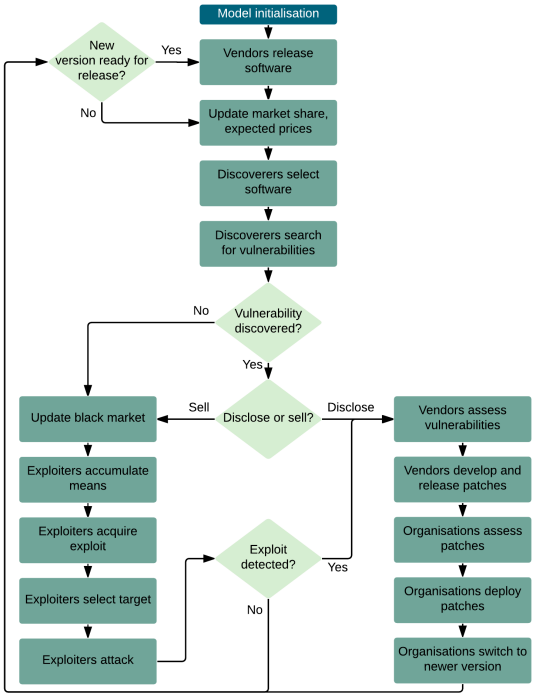

Fig. 2. Flow chart of the agent-based model

Once a vulnerability is discovered and the vendor becomes aware of its discovery, a patch can be developed and released, depending on the vendor's policy. The secure development capability of the vendor corresponds to the probability that more vulnerabilities are introduced in the software when released. In addition, it determines the probability that a patch introduces a new vulnerability. The patch development and patch release procedures are based on [15, 8, 38]. The patch development policy corresponds to probability and speed of patch development. In case of urgency, patch development times can be decreased by a factor three. Table 1 shows the parameter values corresponding to the different levels of both properties. Apart from the average patch development time, these are heuristic values and not based on empirical data. The final vendor property is the patch release policy. There are three release policies; the first is "continuous," which means that a patch is released as soon as it has been developed. The second is "periodical," which means that patches are released every thirty days. The final policy, "semi-periodical," is a combination of the two; patches are release monthly unless multiple attacks are reported, in that case a patch will be released as soon as it is ready.

The final procedure discussed here is the patch deployment by the organisations. The decision to deploy a patch is modelled according to [10] and based on the severity level and reported attacks. If the decision is made to implement the patch, it goes through a testing phase. When the testing phase is completed, the patch is deployed over all the organisation's systems. The security strategy can either be proactive or reactive. The proactive strategy results in a total patch deployment time between 15 – 135 days, depending on the severity, the time it takes to test the patch, and the number of systems within the organisation. The reactive strategy results in an deployment time between 20 – 180 days.

Table 1. Parameter values for different vendor properties

Property	***Low***	***Medium***	***High***
Secure development capability			
Additional vulnerabilities at time of release	20%	10%	0%
Chance that a patch introduces a new vulnerability	100%	66%	33%
Patch development policy			
Chance that a known vulnerability is patched	80%	85 %	90%
Average patch development time	65 days	40 days	15 days

4.2 Expected influence on the attack frequency

The agent-based model is used to test what kind of behaviour of the vendors and organisations influences the number of attacks as a result of vulnerability discovery. The behaviour of the individual agents is determined by their properties. The model includes three properties for the vendors and one property for the organisations. The

three properties for the vendors are their *secure development capabilities*, *patch development policy*, and *patch release policy*. The main property of the organisations is their *security strategy*, which determines if and how fast a patch is deployed throughout the organisation. The expected influence of each of the properties on the attack frequency is summarised in table 2.

Table 2. Hypothesis on the influence of different agent properties on the attack frequency

Property	***Agent***	***Expected influence***
Secure development capabilities	Vendor	Large
Patch development policy	Vendor	Large
Patch release policy	Vendor	Small
Security strategy	Organisation	Large

First, it is expected that the vendor's *secure development capabilities* have a large influence on the attack frequency, as this directly influences the number of vulnerabilities in the software, and thus, increases the difficulty of discovering new vulnerabilities. Second, it is expected that the *patch development policy* of the vendor will have a large impact on the number of attacks, as this directly influences how long an exploit can be used against the organisations. Third, it is expected that the vendor's *patch release policy* only has a small influence on the number of attacks on the organisations that use the software, because the time difference between releasing patches for the different policies is relatively small. Regarding the organisation's *security strategy*, it is expected that organisations with shorter patch deployment times are vulnerable for shorter amounts of time, and are therefore less likely to be attacked compared to organisations with a longer patch deployment time.

5 Results

The first property examined here, is the vendor's ability to develop secure software. The results from this analysis are depicted in figure 3. It shows the distribution of the combined number of attacks on the organisations that use software from a vendor for different secure development capabilities. Each attack is performed by an exploiter, who selects an organisation based on available exploits and value. The simulation records the attacks over a period of 1500 days. The attack frequencies should be interpreted relative to each other, since the model is not calibrated to represented real life values.

The box plot in figure 3 shows that the median of the attack frequency distributions for each of vendor's secure development capabilities do not show a large difference. The probability density function in figure 3 shows that the attack frequencies for each of the properties are not normally distributed. The Shapiro-Wilks Normality Test confirms that the attack frequencies are not normally distributed (with $p < 2.2 \cdot 10^{-16}$). Therefore, a Kruskal-Wallis Test is applied, which does not

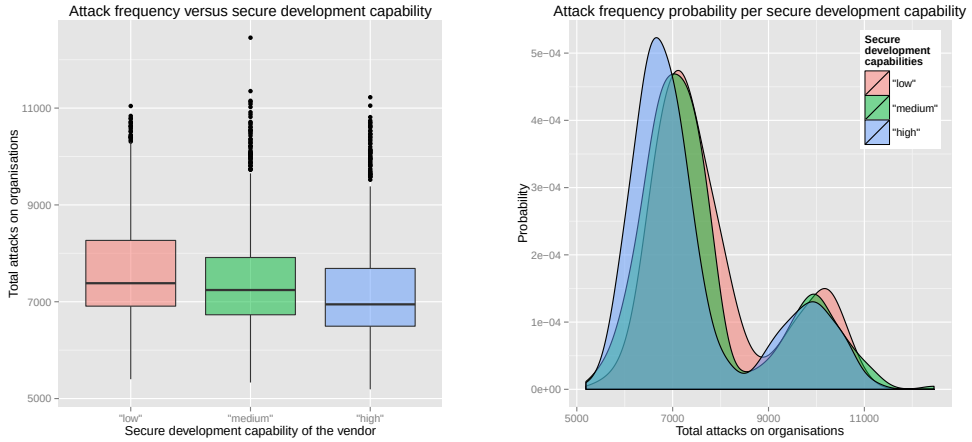

Fig. 3. Attack frequency distribution for different levels of secure development

assume a normal distribution and can test more than two populations. The Kruskal-Wallis test shows (with $p = 6.2 \cdot 10^{-10}$) that there is a significant difference between the different secure development capabilities. However, as can be seen in figure 3, the relative differences in the median of each group are very small. Therefore, it cannot be said to have a large influence based on the relative difference.

This is in contrast to the hypothesis and falsifies the assumption that developing secure software has a large influence on the number of cyberattacks. The results also show a surprising bifurcation, which is most visible in the probability density function in figure 3. The distribution shows two modalities, one around 7000 attacks and the other around 10,000 attacks. The cause can be found in different combinations of vendor properties, which is discussed at the end of this section.

The second set of results focusses on the vendor's patch development policy. The results from this analysis are shown in figure 4. It shows the distribution of the combined number of attacks on the organisations that use software from a vendor for different patch development policies.

Figure 4 shows that vendors that are less likely to develop a patch and take a long time to develop it have a much higher likelihood of exposing the organisations that use their software to higher attack frequencies. The Kruskal-Wallis test shows that the difference in attack frequencies for the different vendor patch development policies, is significant with $p < 2.2 \cdot 10^{-16}$. Looking at the differences between the probability distributions in figure 4, a clear difference is observed between the "high" patch development policy and the "low" and "medium" patch development policies. Both the "low" and "medium" patch development policies show two modalities. One around 7000 attacks and the other around 10,000 attacks. The "high" patch development policy, on the other hand, only has one modality around 6500.

The results from this simulation indicate that the patch development policy has a relatively small influence on the number of attacks on the users of the software, as is shown by the medians in figure 4. This means that the chance that a patch

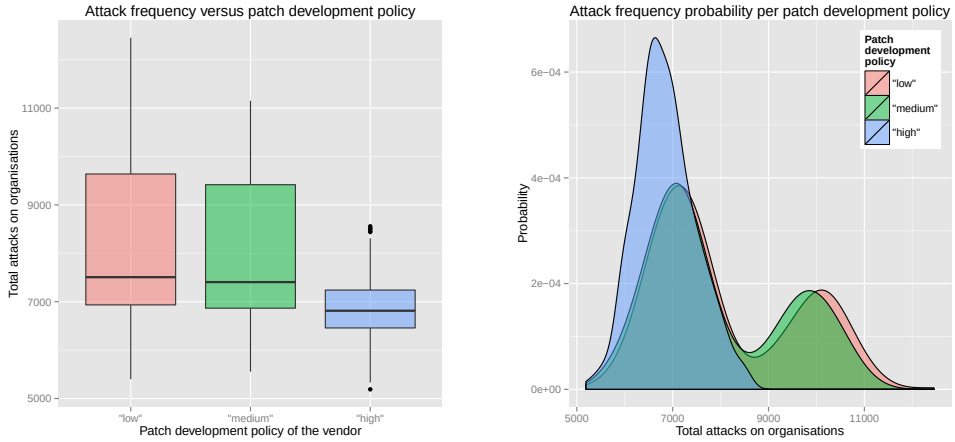

Fig. 4. Attack frequency distribution for different patch development policies

will be developed by the vendor and the speed of the development, do not have a large influence on the risk of large organisations in cyberspace. However, the results do show an interesting difference between the "high" patch development policy and the "low" and "medium" patch development policies. The differences, as shown in figure 4, lead to the conclusion that vendors that do respond quickly to an emerged vulnerability, can reduce the chance of relatively high attack frequencies.

The final property of the vendor that is analysed is the patch release policy. The results from this analysis are shown in figure 5. It shows the distribution of the combined number of attacks on the organisations that use software from a vendor for different patch release policies.

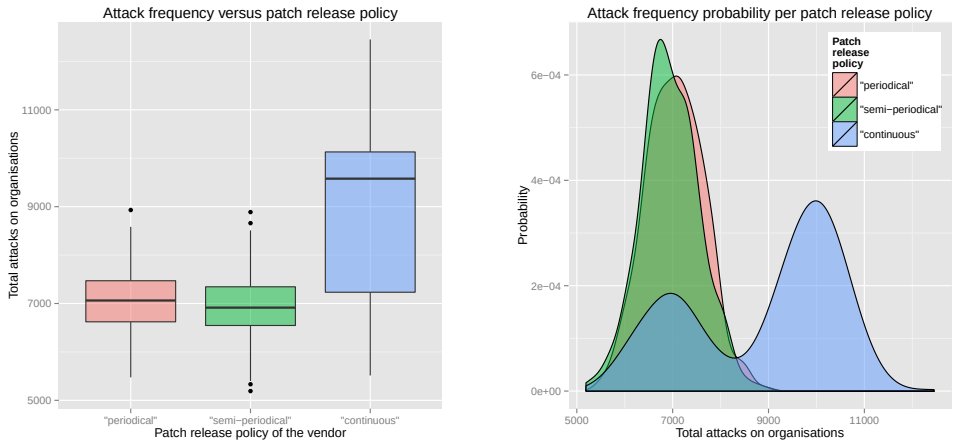

Fig. 5. Attack frequency distribution for different patch release policies

Figure 5 shows some surprising results. Continuously releasing patches increases the number of attacks on the organisations, whereas the differences between periodical and semi-periodical release policy is negligible. The Kruskal-Wallis test shows that the difference in attack frequencies for the different vendor patch release policies, is significant (with $p < 2.2 \cdot 10^{-16}$). Not only is the difference significant, the absolute difference is also large. These results are contrary to what one might expect, as the continuous release policy would enable organisations to deploy patches faster. There is, however, an important underlying dynamic that coincides with the release of a patch. Releasing a patch signals the presence of a vulnerability to the entire Internet. Finding vulnerabilities is difficult, whereas developing an exploit is relatively easy. Releasing a patch for a vulnerability that might only be known to the vendor or a small number of exploiters, becomes public knowledge to all exploiters, which opens the door to a huge increase in the number of attacks. This is exactly the behaviour that is observed in these results. The continuous release of patches does not only provide the organisations with the ability to patch a vulnerability, it also enables a serious increase in the number of attacks if the organisations fail to deploy the patch on a short notice.

The bifurcation between the attack frequencies below 8500 and those above can be explained by a combination of two properties of the vendors. Figure 3 shows that attack frequencies above 8500 occur for each level of secure development capabilities. The first differences in the higher attack frequencies become visible in figure 4. This shows that these higher attack frequencies only occur for "low" and "medium" patch development policies, while a "high" patch development policy shows no attack frequencies above 8500. The most clear difference between the higher and lower attack frequencies becomes visible in figure 5, since it shows that the higher attack frequencies above 8500 only occur with a continuous patch release policy. This leads to the conclusion that the combination between a "low" or "medium" patch development policy and a continuous patch release policy are a worst case combination, which results in an increased number of attacks on the organisations. The risk of higher attack frequencies for different combinations of the patch development time and patch release policy, are shown in table 3. Furthermore, the results show that not the number of vulnerabilities, but the way patches are released, has the largest influence on the number of attacks. This indicates that knowledge diffusion on vulnerabilities has major influence on the risk of large organisations in cyberspace.

Table 3. The likelihood of the organistations being attacked through vulnerabilities as a result of different vendor properties.

	Patch release policy		
Patch development policy	Continuous	Semi-periodical	Periodical
Short development time	*Medium risk*	*Lower risk*	*Medium risk*
Medium development time	*Higher risk*	*Medium risk*	*Medium risk*
Long development time	*Higher risk*	*Medium risk*	*Medium risk*

The final hypothesis concerned the influence of the organisation's security strategy. Given that the focus of this paper is primarily on cyberattacks enabled by software vulnerabilities, this property primarily determines how long it takes to deploy a patch within the organisation. Figure 6 shows how a shorter deployment time (proactive) and a longer deployment time (reactive) influence the number of attacks.

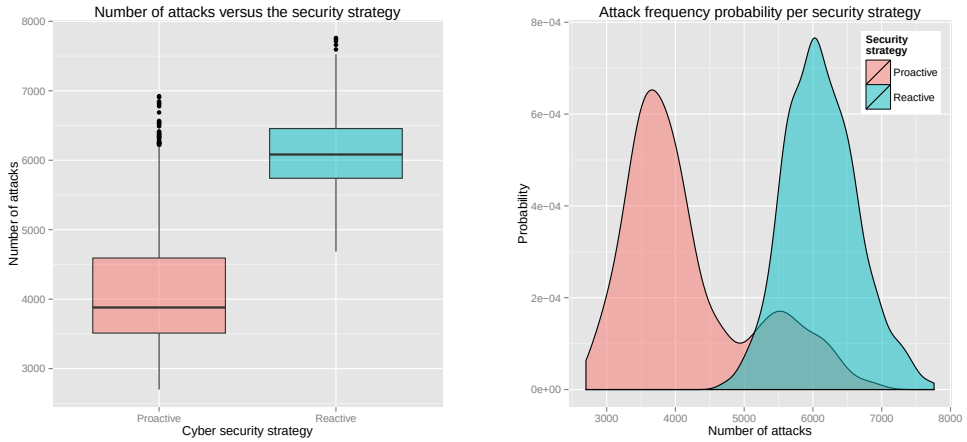

Fig. 6. Attack frequency distribution for different patch release policies

The results show significant differences in the median attack frequencies between the different security strategies. This was to be expected, since organisations with proactive security strategies have an average patch deployment time of 40 days, whereas the average deployment time of organisations with a reactive security strategy is 100 days. This difference is large. However, it is realistic as deployment times for large organisations can differ significantly and range up to several months. The results show that the average difference in patch deployment of 60 days results in a 55% increase of the median attack frequency. This confirms the initial hypothesis and shows that the organisations themselves have significant influence on their own risk exposure in cyberspace. An interesting observation is the multimodal behaviour of the attack frequencies for proactive security strategies, as shown in figure 6. This might be caused by the properties of the vendors that supply the software that is used by the organisations. The organisation's ability to apply patches on a short notice reduces the risk for those organisations of being exposed to the large increase in the number of attacks that follow the release of a patch. Based on these results it can be argued that the organisations themselves have a significant influence on their risk exposure.

6 Conclusion and discussion

The results showed how the attack frequencies vary between different properties of the vendors and organisations. This final section discusses the validity of the model, the conclusions that can be drawn from these results, and their implications.

6.1 Validity and improvements

The results should be interpreted with the model's limitations and the extent of its validity in mind. Based on the guidelines and recommendation for validating agent-based models by [27, 31], three types of validation methods are applied to assess the validity of the model; historic replay, comparison with existing literature, and face validation by experts. The historic replay compared the rate at which vulnerabilities are discovered in the model, with historic data on the rate of discovery for several Windows operating systems. The results only showed differences during the end of the lifecycle of the software, in the sense that the rate of discovery in the model declined faster than the empirical data. However, this difference should not affect the validity of the results, since the simulations focus on the first years after the release. The second test compared the behaviour regarding the vulnerability discoveries with [4, 5] and showed that the model follows that same s-curve. In addition, the attack frequencies as a results of a patch release were compared with [9]. The increase in attack frequencies and the eventual decline showed a similar increase in the attack frequency, as well as the decline of the attack frequency over time. For the face validation, two experts with knowledge and experience of software development, patch management, and CISO-level decision making, were interviewed. Despite minor remarks on the relatively simple representation of the software and the absence of the options of public disclosure, no concerns were raised regarding the validity of the model.

Although the model covers a wider scope of the vulnerability ecosystem and includes more detail compared to existing work, both the scope and the level of detail can be improved. One of the main limitations of this model is the simplified representation of the markets for vulnerabilities, compared to [2]. However, this remains a challenge, since very little is known about the inner workings of especially the black market. A second limitation is the limited diversity of both the discoverers and the exploiters. Given the large range of threat actors and the varying levels of skills, means, and professionalism, the two types of exploiters in the model are an abstract representation of reality. The abstract representation of the actual attack and the response of the organisation is a third limitation of the model. If one would want to take closer look at the interactions between exploiters and organisations, a more realistic mechanism is required, which includes different stages of the attack process and different security mechanisms. A final, although important, limitation of the model is the representation of software itself. A representation of the software which distinguishes different parts of the code, might prove valuable for further use of the model, as it enables the incorporation of more technical characteristics of the software, and allows for differentiation between different types of software.

6.2 Conclusions on the results

Limited research has been done on the risk induced by vulnerabilities on large organisations. This paper provides a first attempt to capture a broad perspective on the vulnerability ecosystem in an agent-based model, and explore vulnerability induced attack frequencies. The model shows that the vendor's ability to develop secure software only has a limited influence on the number of attacks on the organisations. Far more important is how long it takes for the vendor to develop a patch and the timing of releasing a patch. In particular, continuously releasing patches can have a large influence on the number of attacks, when combined with longer patch development times. It is even more important that organisations deploy the patches as quickly as possible. The results show that an average difference of 60 days in the time it takes to deploy a patch, results in a 55% increase in the average attack frequency. This signifies the responsibility of the organisations themselves when it comes to minimising their risk exposure as a result of software vulnerabilities. In addition, the model shows that not necessarily the number of vulnerabilities, but rather the diffusion of the knowledge on vulnerabilities has a strong influence on the resulting attacks. The following points summarise the key insights gained from the agent-based model.

Key insights from the model

- The vendor's ability to develop secure software, and thus minimising the number of vulnerabilities in the software, only has a limited effect on the number of attacks.
- Continuously releasing patches increases the number of attacks, whereas releasing patches periodically unless a vulnerability causes an unusual high risk, results in significant less attacks.
- Not necessarily the number of vulnerabilities, but rather the diffusion of the knowledge on vulnerabilities has a strong influence on the resulting attacks.

6.3 Implications and recommendations

The results show that the response by both the vendor and organisations have a significant influence on the attack frequency. The model shows that not necessarily the number of vulnerabilities, but rather the diffusion of the knowledge on vulnerabilities has a strong influence on the resulting attacks. This is in line with existing observations [9, 11]. However, this research goes a step further by assessing which combination of properties of the vendor has the most influence on the attack frequency. The results indicate that the vendor's release policy can result in a major increase in the attack frequency. In particular, the combination of a continuous release policy with a long average patch development time can result in a surge in the attack frequency. In general, a periodical or semi-periodical patch release policy has a lower risk of resulting in high attack frequencies. This effect is reinforced by the organisation's (in)ability to apply patches. The reason why patch deployment can take a long time can mainly be attributed to a limited capacity for testing and deployment within the organisation [16]. However, the results show that there is much to be gained if organisations improve their capacity to enable faster patch deployment.

The observation that the diffusion of the knowledge on vulnerabilities has a strong influence on the resulting attacks shows that public disclosure and bounty programmes might not be as effective as is currently assumed. They increase the number of patches, and as a result, put more strain on the organisation's capacity to deploy patches. Bounty programmes reveal an initial peak in the number of discovered vulnerabilities in new software, after which, vulnerabilities are discovered at a stable rate [40]. Recent work shows that discovery rates for bounty programmes do slowly decrease over time. However, this seems to be the result of discoverers moving to newer software, which offers quicker rewards due to the presence of more easy to find vulnerabilities [28]. Their findings, together with newly introduced vulnerabilities as a result of constantly changing code [33, 30], show that depleting the number of vulnerabilities in software is difficult. This is an additional argument why public disclosure and bounty programmes might not be as effective as is currently assumed, as there is no evidence that they are depleting the pool of vulnerabilities, while simultaneously put more pressure on the organisation's capacity to deploy patches. This points towards the need for more focus on vulnerabilities that are actually being exploited, instead of trying to patch them all.

Recommendations

- Vendors should only use continuous patch release policies, if the organisations to which they provide the software, are capable of deploying the patch in a timely manner.
- Vendors should put more effort in developing capabilities to determine and predict which vulnerabilities are likely to be exploited, instead of patching all known vulnerabilities.
- Organisations should improve their capabilities to deploy patches in order to reduce unnecessary exposure to cyberattacks.

Future research

- In order to support vendors in determining which vulnerabilities should be patched and which should be kept from becoming public knowledge, methods to analyse and predict which vulnerabilities are likely to be exploited should be improved.
- A balance has to be found between on the one hand patching a vulnerability and risking large scale exploitation. On the other hand, not releasing a patch leaves each user unnecessarily vulnerable. Further research should analyse the risk of both options and determine how such a decision should be made.
- Agent-based modelling has proven to be valuable for addressing the problems in this paper. Research regarding cybersecurity-related problems can benefit from agent-based modelling in case empirical data is scarce. The model used in this paper can be altered and improved to address various related questions.

References

1. Ablon, L., Libicki, M.C., Golay, A.A.: Markets for Cybercrime Tools and Stolen Data. RAND Corporation (2014)

2. Algarni, A., Malaiya, Y.K.: Software Vulnerability Markets: Discoverers and Buyers. International Journal of Computer, Information, Systems and Control Engineering 8(3), 449–459 (2014)
3. Alhazmi, O.H., Malaiya, Y.K.: Modeling the vulnerability discovery process. In: Proceedings of International Symposium on Software Reliability Engineering, ISSRE. pp. 129–138 (2005)
4. Alhazmi, O.H., Malaiya, Y.K.: Quantitative vulnerability assessment of systems software. In: Proceedings of 51st annual reliability and maintainability symposium. pp. 615–620. Alexandria, VA (2005)
5. Alhazmi, O.H., Malaiya, Y.K., Ray, I.: Measuring, analyzing and predicting security vulnerabilities in software systems. Computers and Security 26(3), 219–228 (2006)
6. Anderson, R.: Security in Open versus Closed SystemsThe Dance of Boltzmann, Coase and Moore. In: Conference on open source software: economics, law and policy. pp. 1–15. Toulouse, France (2002)
7. Arbaugh, W.A., Fithen, W.L., McHugh, J.: Windows of vulnerability: A case study analysis. Computer 33(12), 52–58 (2000)
8. Arora, A., Krishnan, R., Telang, R., Yang, Y.: An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure. Information Systems Research 21(1), 115–132 (2010)
9. Arora, A., Nandkumar, A., Telang, R.: Does information security attack frequency increase with vulnerability disclosure? An empirical analysis. Information Systems Frontiers 8(5), 350–362 (2006)
10. Beres, Y., Griffin, J., Shiu, S., Heitman, M., Markle, D., Ventura, P.: Analysing the performance of security solutions to reduce vulnerability exposure window. In: Proceedings of Annual Computer Security Applications Conference, ACSAC. pp. 33–42 (2008)
11. Bilge, L., Dumitras, T.: Before We Knew It: an Empirical Study of Zero-Day Attacks in the Real World. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security – CCS'12. pp. 833–844 (2012)
12. Böhme, R., Moore, T.: The Iterated Weakest Link. IEEE Security & Privacy Economics (1), 53–55 (2010)
13. Bonabeau, E.: Agent-based modeling: methods and techniques for simulating human systems. In: Proceedings of the National Academy of Sciences. vol. 99, pp. 7280–7287 (2002)
14. Browne, H.K., Arbaugh, W.a., McHugh, J., Fithen, W.L.: A trend analysis of exploitations. In: Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on. pp. 214–229 (2001)
15. Cavusoglu, H., Cavusoglu, H., Zhang, J.: Security Patch Management: Share the Burden or Share the Damage? Management Science 54(4), 657–670 (2008)
16. Cavusoglu, H., Zhang, J.: Economics of Security Patch Management. In: Workshop on the Economics of Information Security (WEIS) (2006)
17. Finifter, M., Akhawe, D., Wagner, D.: An empirical study of vulnerability rewards programs. Proceedings of the 22nd USENIX Security Symposium pp. 273–289 (2013), https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_finifter.pdf
18. Frei, S., Schatzmann, D., Plattner, B., Trammell, B.: Modeling the Security Ecosystem – The Dynamics of (In)Security. In: Moore, T., Pym, D., Ioannidis, C. (eds.) Economics of Information Security and Privacy, pp. 79–106. Springer US (2010)
19. Gorodetski, V., Kotenko, I.: The multi-agent systems for computer network security assurance: frameworks and case studies. In: Proceedings of IEEE International Conference on Artificial Intelligence Systems. pp. 297–302. Liniya (2002)

20. Joh, H., Malaiya, Y.K.: Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. international conference on security and management (SAM) (1), 10–16 (2011)
21. Khoo, W., Aloteibi, S., Anderson, R.J., Meeks, M.: Hunting for vulnerabilities in large software: the OpenOffice suite. Cambridge University press 1(3), 87–100 (2010)
22. Kim, J., Malaiya, Y.K., Ray, I.: Vulnerability discovery in multi-version software systems. In: Proceedings of IEEE International Symposium on High Assurance Systems Engineering. pp. 141–148 (2007)
23. Kotenko, I.V.: Agent-based modeling and simulation of cyber-warfare between malefactors and security agents in Internet. In: Proceedings of 19th European Conference on Modelling and Simulation (2005)
24. Kotenko, I.V., Konovalov, A., Shorov, A.: Agend-based Modeling and Simuation of Botnets and Botnet Defense. In: Proceedings of Conference on Cyber Conflict. pp. 21 – 44 (2010)
25. Kwon, J., Johnson, M.E.: An Organizational Learning Perspective on Proactive vs . Reactive Investment in Information Security. In: Workshop on Economics of Information Security (WEIS). pp. 1–29 (2011)
26. Libicki, M.C., Ablon, L., Webb, T.: The Defender's Dilemma. RAND Corporation, Santa Monica, California (2015)
27. Louie, M.A., Carley, K.M.: Balancing the criticisms: Validating multi-agent models of social systems. Simulation Modelling Practice and Theory 16(2), 242–256 (2008)
28. Maillart, T., Zhao, M., Grossklags, J., Chuang, J.: Given Enough Eyeballs , All Bugs Are Shallow ? Workshop on the Economics of Information Security (WEIS) (2016)
29. Miller, C.: The Legitimate Vulnerability Market Inside the Secretive World of 0-day Exploit Sales. In: Workshop on Economics of Information Security (WEIS). pp. 1–10 (2007)
30. Neuhaus, S.: Software Security Economics : Theory, in Practice. In: Workshop on Economics of Information Security (WEIS). pp. 1–21 (2012)
31. Nikolic, I., van Dam, K.H., Kasmire, J.: Practice. In: Van Dam, K.H., Nikolic, I., Lukszo, Z. (eds.) Agent-based modelling of socio-technical systems, pp. 73–135. Springer Science & Business Media, Dordrecht, 9 edn. (2013)
32. NIST: Risk Management Guide for Information Technology Systems Recommendations of the National Institute of Standards and Technology. Tech. Rep. July, National Institute of Standards and Technology (2002)
33. Ozment, A.: Improving Vulnerability Discovery Models - Problems with Definitions and Assumptions Categories and Subject Descriptors. In: 2007 ACM workshop on Quality of protection. pp. 6–11. ACM (2007)
34. Pidd, M., Seibers, P.O., Macal, C.M., Garnett, J., Buxton, D.: Discrete-event simulation is dead, long live agent-based simulation! Journal of Simulation 4(3), 204–210 (2010)
35. Roumani, Y., Nwankpa, J.K., Roumani, Y.F.: Time series modeling of vulnerabilities. Computers & Security 51(0), 32–40 (2015)
36. Schryen, G.: A comprehensive and comparative analysis of the patching behavior of open source and closed source software vendors. IMF 2009 - 5th International Conference on IT Security Incident Management and IT Forensics - Conference Proceedings pp. 153–168 (2009)
37. Takanen, A., Demott, J., Miller, C.: Fuzzing for software security testing and quality assurance. Artech House, Norwood, MA (2008)
38. Temizkan, O., Kumar, R.L., Park, S., Subramaniam, C.: Patch Release Behaviors of Software Vendors in Response to Vulnerabilities: An Empirical Analysis. Journal of Management Information Systems 28(4), 305–338 (2012)

39. Wilensky, U.: NetLogo (1999), http://ccl.northwestern.edu/netlogo/
40. Zhao, M., Grossklags, J., Liu, P.: An Empirical Study of Web Vulnerability Discovery Ecosystems. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1105–1117. ACM (2015)