4 SYSTEM ANALYSIS

Chapter 3 analyzed the occurrence due to a vulnerability in Citrix software. This occurrence was not an isolated one. The chapter also analyzed similar incidents where vulnerabilities in software led to security breaches at organizations. In some cases, this directly impacted people's security and safety. This illustrates that vulnerabilities in software are not isolated incidents. They are symptoms of a larger problem. The occurrences reveal a common thread: organizations and the people who depend on them are exposed to digital unsafety. Unknowingly they use software that is vulnerable. In many cases, warnings do not reach them and organizations do not always have the resources to remedy the vulnerability.

Chapter 4 analyzes the problem at the system level. In so doing, we distinguish between the process in which software is developed; the process in which organizations select certain software to purchase and put into use; and the processes that take place after a vulnerability in the software is found (incident response). In addition, we address how stakeholders, such as manufacturers, organizations that use software, and the government as policymaker, learn from digital incidents. We also address the role that the international context plays in managing insecurity and unsafety due to vulnerabilities in software.

4.1 Producing and releasing software on the market

Software fulfils a crucial role in the functioning of digital systems within organizations. Software is for example used to gain access to the company network from home, and as such forms the link between the internal and external network (Internet). Products of this kind therefore play an essential role in safeguarding cybersecurity.

Vulnerabilities are always inherent in software products, some of which lead to major safety risks. These risks are real and there have already been several examples, with disruptive consequences for public services. A vulnerability in a software product for example (indirectly) led to serious disruptions in service provision by Dutch municipalities (it was no longer possible to pay supplementary benefits to local residents) and a hospital (patients were no longer able to access their personal files and no information could be exchanged with other hospitals). Chapter 3 described examples of vulnerabilities in such products and their consequences. In this section we discuss how it is possible that software contains vulnerabilities and how manufacturers estimate the risk of these vulnerabilities and their consequences, and take measures to prevent or limit those consequences.

In 4.1.1 we describe the factors that explain why vulnerabilities can emerge in software and we describe the incentives that affect those factors. In 4.1.2 we then outline the measures taken by manufacturers to discover vulnerabilities, both before and after the software is released, the difficulties this process involves, and the dilemmas the manufacturers face. Finally, in 4.1.3 we consider the patching of vulnerabilities and the manufacturer's role in incident response.

4.1.1 Preventing vulnerabilities in the lifecycle of software

Vulnerabilities can arise at any point in the lifecycle of a software product. For example, a vulnerability can emerge during the initial development of a new product, but equally during the renewal or improvement of existing software, in the form of an upgrade, or sometimes even as a consequence of fixing another vulnerability. Interviews with manufacturers and literature studies show that a number of factors contribute to the emergence of vulnerabilities during the lifecycle of a product. Below we discuss a number of factors.

Software products have a history

The first factor relates to the history of the development of software products, which is sometimes long and complex. Over time, manufacturers add new functionalities to existing software packages on multiple occasions, therefore building on an existing product. In certain cases, the original code of the software package (the foundation) is more than twenty years old. Changing needs and increasing digitization in society mean that software is taking on a different role. As a result, a software product is never finished. Manufacturers respond to this time and again by using existing platforms and adding extra functionalities, or reusing existing components.

Because manufacturers repeatedly add additional functionalities, the number of lines of code increases, and the software becomes more complex.[108] It is not uncommon for a software product to consist of more than one million lines of code.[109] Interviews and literature studies show that even with an extended framework for product development, safely maintaining such huge quantities of code is a significant task. Manufacturers therefore oftentimes restrict themselves to fixing the specific vulnerability as published in the CVE.[110] Dealing with the underlying cause in the foundation of the product (programming language, components, architecture) can require the complete rebuilding of the product. Manufacturers consider this to be too costly. Large software companies are often stock exchange-quoted companies and financial considerations play a role. However, the development history of software sometimes means that a product has grown in such a way that fixing a vulnerability is nothing more than tackling symptoms. In reality, a complete revision of the basis of the product may be needed to truly solve the (safety) problem.

108 https://www.extremetech.com/computing/259977-software-increasingly-complex-thats-dangerous.
109 https://www.informationisbeautiful.net/visualizations/million-lines-of-code/.
110 Common Vulnerabilities and Exposures. A public list of known weaknesses in software. The list appears on https://cve.mitre.org. (Source: Cybersecurity Alliantie, *Cybersecurity Woordenboek*, 2019, https://www.cybersecurityalliantie.nl/binaries/cybersecurityalliantie/documenten/publicaties/2019/09/30/cybersecurity-woordenboek/VCNL-Woordenboek-2eDruk-webversie-Final-2.pdf).

Programming language
A second explanatory factor that can also influence the emergence of vulnerabilities is the programming language used. The programming language currently most commonly used (C/C++) is recognized as being 'unsafe', because it allows programmers considerable leeway to make mistakes.[111]

Manufacturers have access to a series of general tools for eliminating whole classes of vulnerabilities, or mitigating their effects. Around half of the security breaches over the past few years have been related to vulnerabilities in memory security, that can be rectified by writing code in more secure languages such as Rust, or by subjecting the existing C/C++ code to verification tools.[112]

According to research, it is unattractive for manufacturers to protect software development against vulnerabilities: it makes the software slow, and during the programming process, the programmers receive so many (sometime erroneous) error messages that they switch off the security system.[113]

It is also not possible with all programming languages to use tools to detect vulnerabilities during the development process.[114] In the Citrix case, for example, the fact that the programming language Perl was barely supported if at all by these scanning tools played a clear role. See also 4.1.2 on what manufacturers do to discover vulnerabilities, and the obstacles they come across.

Use of standard components
The third factor is the use of standard components. When developing software, manufacturers make regular use of existing (open source) software components. Examples are the Apache and NGINX HTTP server, that are often used as the basis for software with web functionality. A manufacturer can also reuse components from their own existing software or from software that was previously made by an acquired company..

By reusing other components and the associated code, the manufacturer also incorporates all (undiscovered) vulnerabilities contained in that code.[115] Once the code has been integrated in the developer's own package, it takes a great deal of effort to

111 The basis for the SSL VPN (a virtual private network that uses the SSL or TLS protocol) and much other software is C/C++. Programming languages like C enable programmers to write code at a higher level of abstraction. This refers to the proximity of the programming language to the hardware. At a higher level of abstraction, developing software becomes simpler and more understandable than at a lower level, whereby more specific machine instructions are needed. However, that too can lead to errors. C is a programming language which is recognized as being 'unsafe', because in this language, working memory management is carried out manually (Kroes, T., *How to Keep Your Memory Safe and Your Software Fast,* 2020; AG Connect, *Einde van de oneindige reeks softwarefouten in zicht,* 2021). This is error sensitive and the majority of SSL VPNs use their own additions to existing programming languages. This can lead to simple memory errors; the most common source of software bugs and an important area of attack for attackers (see https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues). Nonetheless, C remains one of the most widely used programming languages.

112 Anderson, R., *Security Engineering*, 2020.

113 Kroes, T., *How to Keep Your Memory Safe and Your Software Fast,* 2020 https://research.vu.nl/en/publications/how-to-keep-your-memory-safe-and-your-software-fast

114 Tjong Tjin Tai, E. and Knoops, B., Duties of care and diligence against cybercrime *(Nederlands Juristenblad 24-04-2015*, volume 16), 2015.

115 AG Connect, *Veel kritieke lekken door open source in standard apps, (numerous critical leaks caused by open source in standards apps),* 2021. https://www.agconnect.nl/artikel/veel-kritieke-lekken-door-open-source-standaard-apps

update the underlying component in the event of a vulnerability. By that stage, the software package is after all dependent on a particular version of the component. In addition, manufacturers do not always have access to the relevant knowledge to be able to update components produced by others.[116]

Architecture
The fourth factor that contributes to the presence of vulnerabilities relates to the situation when the different layers that make up the architecture of the product are mutually inconsistent. For the functioning of the software, it is essential that the various components that make up the software match successfully. The matching of the various components must have been achieved in a controlled manner, under the supervision of a person with considerable experience, and sufficient knowledge and who has a major stake in the security of the product.[117]

Configuration
A last factor, which does not necessarily contribute to the emergence of vulnerabilities, but can limit their impact, is the way in which the software is configured by the manufacturer (the default settings). This includes which rights are granted to different types of users, how these rights are set by default, and whether it is possible as a customer to restrict these rights.

A range of factors contribute to the emergence of vulnerabilities during the lifecycle of a product. In many cases, existing products undergo further development, making the software increasingly complex. The programming language used can also contribute to the occurrence of errors, and the use of existing components and (inconsistent) layers in the architecture may introduce vulnerabilities.

Whenever (safety) problems are linked to fundamental choices in the product, this can represent an obstacle for the manufacturer in tackling the root of the problem. Such an approach after all requires an investment in the form of money and/or capacity for problem solving. The decision by the manufacturer to instead opt to only fix the vulnerability is explainable, but to truly solve a (safety) problem, it is sometimes necessary to fully revise a product from the base up.

4.1.2 Identifying vulnerabilities during the lifecycle
Manufacturers have established processes for detecting vulnerabilities during the development and use of a product. In this section, we discuss in more detail the measures that manufacturers can take in order to find vulnerabilities, together with the dilemmas they can face.

116 Tsai, O., *Infiltrating Corporate Intranet Like NSA*, 2020. https://i.blackhat.com/USA-19/Wednesday/us-19-Tsai-Infiltrating-Corporate-Intranet-Like-NSA.pdf
117 Anderson, R., *Security Engineering*, 2020.

Action perspective of the manufacturer

Manufacturers detect vulnerabilities by carrying out a series of different tests both before, during and following completion of the development process. For open source software, the source code is openly available to anyone. This means that errors in the code can be unveiled by third parties, even if they are not specifically requested to do so. This is not possible for closed source code, and it is up to manufacturers to take the initiative to carry out an audit.

Manufacturers can be expected to carry out constant security analyses of the entire architecture of the product (see reference framework: the role of the manufacturer and user in chapter 2). Manufacturers use a variety of methods for developing software, for example the Secure Development Lifecycle (SDLC).[118] Part of this involves the manufacturers testing for vulnerabilities at various moments during the development process (during initial development and when releasing patches). These tests are carried out on individual components (unit testing), the integration between components (integration testing) and on the entire product (audit[119] or security code review).

By using automated tools, manufacturers are able to remove more vulnerabilities from software. In this way, they hope to extend the lifecycle of the software. However, the security code reviews of the entire product do not always recognize the type of vulnerabilities relevant in this case. Vulnerabilities are not (always) the consequence of errors in the source code, but may also be the result of integration problems within the product. To detect vulnerabilities of this kind, the manufacturer can also opt to have the product extensively tested for its intended functioning (end-to-end testing). Interviews with manufacturers reveal that for older products, end-to-end testing can be very time consuming, because older products often consist of large volumes of source code.

Manufacturers can also search for vulnerabilities without directly giving third parties access to the source code. Many manufacturers operate bug bounty programmes, according to which in exchange for a reward, ethical hackers search for vulnerabilities in the software. These ethical hackers use manual search methods, firstly aiming their search at easily identifiable vulnerabilities that are the result of fundamental design choices and problems in the integration or configuration.

Many of these bug bounty programmes are open to anyone, but certain manufacturers also opt for a closed variant or decide not to operate any form of bug bounty programme. Manufacturers are sometimes also sent information about vulnerabilities from the bug bounty programmes of other parties, such as suppliers or customers. For example, at the time of the incident Citrix operated only a closed bug bounty programme; the company has however recently launched an open programme.

118 See for example https://owasp.org/www-pdf-archive/Jim_Manico_(Hamburg)_-_Securiing_the_SDLC.pdf

119 Part of the audit performed by the manufacturer is for example threat modelling (identifying threats and mitigating measures) and pen testing (testing for vulnerabilities and attempting to hack into the system). Pen tests can in part be automated, but this test software can also contain vulnerabilities (see for example https://arstechnica.com/gadgets/2021/08/critical-cobalt-strike-bug-leaves-botnet-servers-vulnerable-to-takedown/).

Manufacturers are required to maintain an overview of their customers, those who bought a software product (see reference framework in chapter 2). This enables the manufacturer to warn its customers quickly, in the event of a vulnerability. Not all manufacturers have an up-to-date overview of the customers of their products. This is because products are not always sold directly to the customer; there are often a whole raft of intermediaries. Interviews revealed that certain manufacturers have solved this problem by linking contact details of the customers to their own overview, even if the product is sold via an intermediary. From the safety perspective, it seems obvious that manufacturers have an overview of the customers of a product. However, it may present a dilemma that affects, among other things, the autonomy of the customer. For instance, it is not possible to force customers to register themselves with a manufacturer and to provide transparency on how the system is installed.

A trend that has emerged over the past few years is for manufacturers to migrate their products to the cloud (Software as a Service) in order to improve test capability and to install patches on their customers' systems more quickly. This makes the patching of a product the responsibility of the manufacturer. However, it does involve certain disadvantages for the customer, see section 4.2.

Asymmetry: manufacturer needs to find everything; hackers need just one leak
Manufacturers have to put a lot of time and effort into detecting vulnerabilities, both before and after the software is released. Using such techniques as end-to-end testing, it is possible to remove many vulnerabilities from the software. However, searching for just a single vulnerability takes a great deal of effort. In terms of prevention, manufacturers are already doing everything they can. Problems in software that has been in use for a longer period of time (see the development history in subsection 4.1.1) are unavoidable given the extent of the product and the prevention paradox. It is after all not possible to detect all vulnerabilities.

For their part, attackers attempt to find a vulnerability in a system with different methods, for example a brute-force attack. They sometimes launch their attacks in response to specific clues (for example using information from a CVE), but they regularly also come across a vulnerability by coincidence. Attackers sometimes need just a single leak in order to gain full access to a system. This reveals an imbalance between attacker and defender (manufacturer).

Whereas in the past attackers themselves needed to search the Internet for vulnerable servers (a time-consuming process), the use of services that scan the Internet have made this process far easier.[120] Using scan services of this kind, attackers can easily purchase a list of IP addresses relating to a (just published) CVE. This means that after discovery of a vulnerability attackers have immediate access to a list of potentially vulnerable servers.

120 For example Shodan (https://www.shodan.io), a search engine that scans the Internet and indexes accessible IP address and port combinations. If a server is indexed, then it can be approached over the Internet. This does not automatically mean that the server is also vulnerable. That is something the hacker has to determine.

Relationship between manufacturers and ethical hackers / red teams

Ethical hackers make an important contribution to the identification of vulnerabilities. Bug bounties (earning a reward for reporting a vulnerability) are relevant incentives in this regard. The majority of major manufacturers operate a bug bounty programme[121] that offers ethical hackers an opportunity to earn money by identifying and reporting vulnerabilities. Finding and publishing about a specific vulnerability can also increase the name awareness of a hacker or group of hackers. This mechanism, in combination with the potential financial gain, means that third parties regularly go in search of and subsequently find many vulnerabilities.

At the same time, vulnerabilities increasingly represent a potential attack route (see subsection 4.1.3) and preventing and fixing these vulnerabilities requires tremendous effort on the part of manufacturers (see subsection 4.1.1). In that sense, it would help the manufacturers, and help protect systems if the vulnerabilities were kept secret. It is possible to publish about vulnerabilities without revealing the specifics of a vulnerability. But some manufacturers deliberately choose not to disclose all (information about the presence of) vulnerabilities.[122] This approach, however, is diametrically opposed to the timely disclosure of information about vulnerabilities for mitigating and responding to potential risks. There is a clear incentive for preventing information about vulnerabilities becoming public. Disclosure makes it possible for customers to countermeasure the consequences but at the same time leads to a new security problem: a dilemma.

Parties that discover a vulnerability do not always report their discovery to the manufacturer. Vulnerabilities in software are a tradeable commodity, that is not only reported to the manufacturer (sometimes in return for a reward), but that can also be offered to the highest bidder. For state actors and criminals, obtaining a list of unknown vulnerabilities which they themselves can subsequently exploit can prove attractive.[123] Commercial spyware products are also available for sale. It is unclear whether these products are based on unknown vulnerabilities, and it is also uncertain which parties are offered these products, and for what purpose they are used.[124]

121 For a list of bug bounty programmes, see for example https://www.bugcrowd.com/bug-bounty-list

122 For example Palo Alto, where according to the security researcher that found the vulnerability, no CVE was published about a vulnerability (which had in fact already been repaired by the manufacturer) in GlobalProtect. Source: https://blog.orange.tw/2019/07/attacking-ssl-vpn-part-1-preauth-rce-on-palo-alto.html. It is unclear whether Palo Alto communicated with their customers about the vulnerability through direct channels. The Safety Board was unable to verify this because Palo Alto did not respond to our requests to cooperate with the investigation.

123 Perlroth, N., *This is how they tell me the world ends: the cyberweapons arms race*, 2021.

124 https://www.wired.com/story/nso-group-hacks-ios-android-observability/
https://www.nrc.nl/nieuws/2021/07/26/de-overheid-moet-stoppen-met-gebruik-van-zero-day-software-a4052412

Ethical hackers are encouraged with rewards to identify and report vulnerabilities in software. As a result, many vulnerabilities are identified. In addition, manufacturers detect vulnerabilities by carrying out a variety of tests. Nonetheless, it is not possible to find *all* vulnerabilities. It is becoming more common for vulnerabilities to form an attack route. Disclosing a vulnerability can help organizations better arm themselves against potential exploitation, but it can also enable attackers to exploit the vulnerability. This is reinforced by the fact that sometimes hackers need just a single leak in order to gain access to a system, also because it is relatively simple for them to find vulnerable servers. This creates a dilemma which in turn reduces overall safety.

4.1.3 The role of vulnerabilities in cyber (in)security

Vulnerabilities are playing an ever growing role

Each year, organizations are exposed to a large and ever growing number of vulnerabilities. In 2020, more than 25,000 vulnerabilities were identified. Of these vulnerabilities, 18,000 were published in 2020 with a CVE number[125] (see Figure 16). Only a small proportion of the number of published vulnerabilities (around 3%) are used to hack organizations and/or individuals. An even smaller proportion (0.5%) are successfully used in practice to launch a widespread attack as described in the security breaches in chapter 3 (see Figure 17). Nonetheless, numbers are growing, and experts warn that we are just seeing the tip of the iceberg.[126]

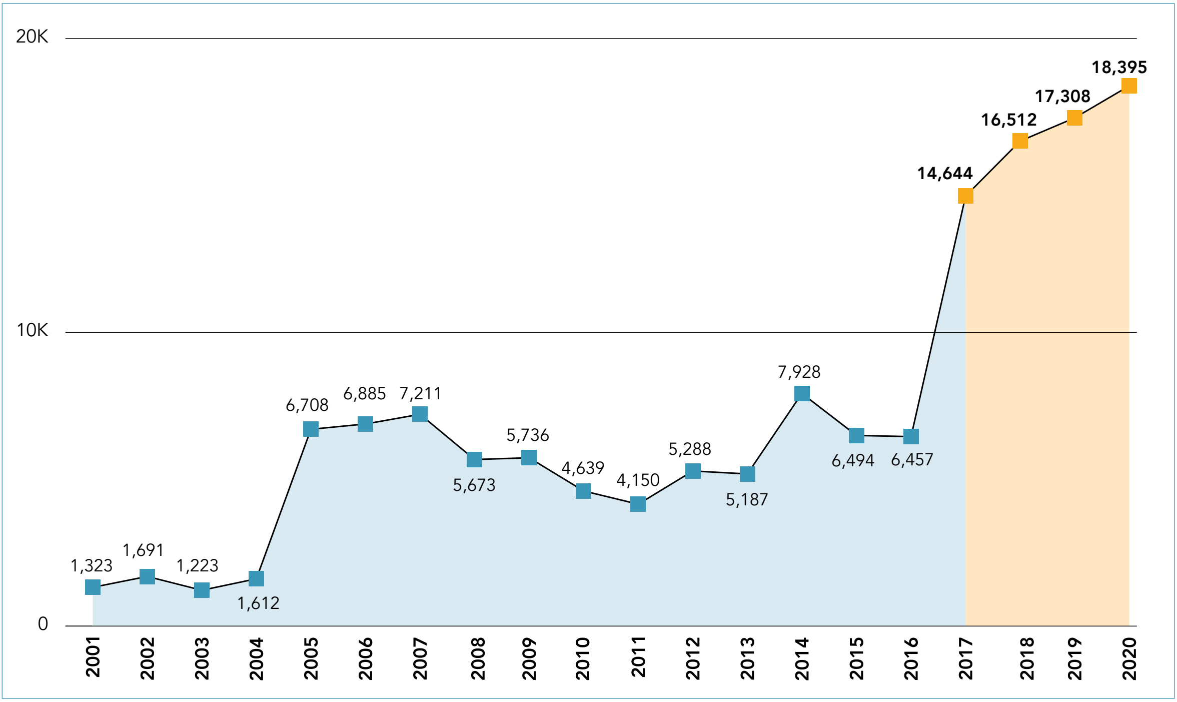

Figure 16: The number of CVE reports per year. (Source: Trend Micro)

125 There are also many vulnerabilities that are fixed by the manufacturer without disclosure. https://vulndb.cyberriskanalytics.com/#statistics

126 AG Connect, *Einde van de oneindige reeks softwarefouten in zicht (End of an infinite series of software errors in sight)*, 2021.

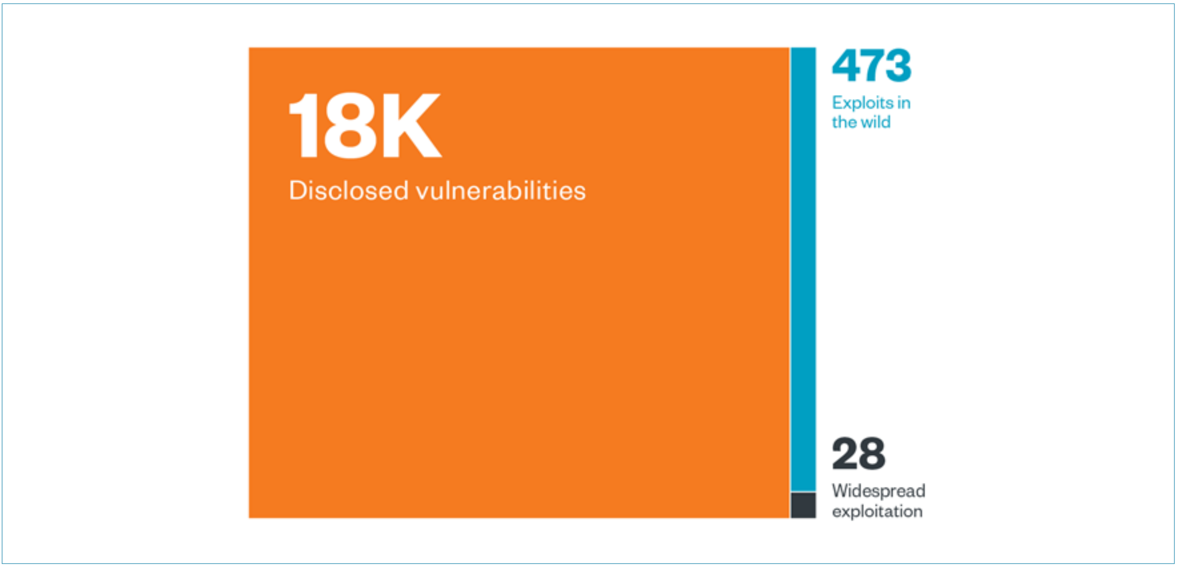

Figure 17: The number of exploits of widespread attacks in relation to the total number of reported vulnerabilities. (Source: Trend Micro)

The consequences of these attacks are also increasing in scale. In its Cyber Security Assessment Netherlands (CSAN) 2020, the Dutch National Coordinator for Security and Counterterrorism (NCTV) warned of attackers searching for weak links in the supply chain, as the next step towards attractive targets and the resultant serious consequences.[127] Whereas in the past a vulnerability in a software package did not automatically result in serious consequences, today they can have far-reaching consequences for the underlying dependent systems, as illustrated by the supply chain attacks using the vulnerabilities in SolarWinds and Kaseya (see section 3.3 for a brief analysis).

In other words, vulnerabilities such as those described in the occurrences investigated by us are playing an ever greater role in cyber-attacks and are increasingly being used by attackers as the starting point to launch an attack.[128] Above all large organizations (such as governments and vital operators) run the risk of being attacked according to this target vector.[129] It has become clear since 2020 that the exploitation of vulnerabilities in software to launch ransomware attacks is an economically attractive method for ransomware gangs.

Growing numbers of widespread attacks using a vulnerability demonstrate the importance of the timely patching of software and/or the mitigation of a vulnerability. The use of software introduces risks. For instance, for organizations it is not always possible to predict which of the vulnerabilities will eventually form a risk for their organization. This depends for example on how easy it is to actively exploit the vulnerability in the software, whether a mitigation is available and how easily it can be

127 NCTV, *Cybersecuritybeeld Nederland 2020*, 2020. https://www.ncsc.nl/documenten/publicaties/2020/juni/29/csbn-2020

128 Modderkolk, H., 'Overheid doet te weinig tegen ransomware' *(Government failing to take sufficient action against ransomware)* (*De Volkskrant*, 4 August), 2021; CISA, Alert (AA21-209A) Top Routinely Exploited Vulnerabilities, 2021

129 Coveware, *Ransomware Attack Vectors Shift as New Software Vulnerability Exploits Abound*, 2021. https://www.coveware.com/blog/ransomware-attack-vectors-shift-as-new-software-vulnerability-exploits-abound

implemented, and the version and configuration of a product. Fixing vulnerabilities by implementing a mitigating measure or installing patches requires an investment by the organization. In most cases, they do not immediately get more security, in return.

For manufacturers and organizations that use the software, prevention, timely mitigation or patching of a vulnerability do not represent the only lines of defence. Section 4.2 considers in more detail the measures that organizations can take to mitigate the safety risks of vulnerabilities in software. Examples are the use of a firewall to restrict access to the network, and the use of redundant hardware and software, so that when a vulnerability is made public, it is possible to switch rapidly to another product.

Problems with patching and mitigating
If a manufacturer has placed software on the market that subsequently turns out to contain a vulnerability, as a rule the manufacturer publishes a patch and advises organizations to patch the software. If no patch is yet available, a manufacturer can also publish a mitigation measure to remove the acute danger. However, patching and mitigating are not always easily implemented solutions.

Patches and mitigations represent a certain degree of risk, too. The effect of a patch or mitigation on software that is already configured and in use cannot always be predicted. Every mitigation and patch can result in (partially) unforeseen consequences, for example for the compatibility of adjacent/connected systems. In certain cases, patches can even cause disruptions or the entire failure of systems.[130] Patches and mitigations can also introduce new errors in the software or introduce vulnerabilities, as for example was the case with the Microsoft patch aimed at solving the problems with the print spooler, which led to problems with printing.[131]

Vulnerabilities in software formed an escalation factor. The occurrences in this investigation are clear illustrations. After the vulnerabilities had become known (for example through the publication of a CVE or a security bulletin), attackers used automated tools to search for servers that had not yet been patched, and to subsequently launch attacks. A mitigation measure can also provide information about how a vulnerability can be exploited. The occurrences in this investigation reveal that this can take place in a period of just a few days (or that the attacks had already been carried out, in the event of a zero day). The publication of a vulnerability can be the lead-up to widespread attacks.

Problems with patching can also arise on the side of the user. Because of the large number of patches published each year, it is for example not always possible to install everything in a timely fashion. Organizations are also not always in possession of an up-to-date overview of which software needs to be patched, they often have limited insight into underlying (vulnerable) components contained in a software package, and they are not always convinced of the necessity of patching. This is discussed in more detail in section 4.2.

130 https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/the-nightmares-of-patch-management-the-status-quo-and-beyond
131 https://www.zdnet.com/article/microsofts-printnightmare-patch-is-now-causing-problems-for-some-printers/

The release of a mitigation measure before a patch is published can be good practice, because following its publication, organizations as a rule implement the measure without delay. In this way, a manufacturer ensures that the end user's software is safe. The disadvantage is that certain organizations then consider patching to be even less necessary.

The number of vulnerabilities in software is growing, as are the consequences of attacks. Vulnerabilities play an increasingly important role in cyberattacks, and can be used by attackers as the starting point for launching an attack. This underpins the importance of timely patching. However, patching and mitigating at the same time pose a risk, because they can lead to disruptions or the introduction of new vulnerabilities. The organization must therefore think through the decision to patch carefully from the perspective of the organization's IT landscape. The publication of a vulnerability can be the precursor to widespread attacks.

4.1.4 Incentives for more secure software

In addition to the more intrinsic factors relating directly to the development process at the manufacturer, other factors relating to regulation and liability also play a role in the emergence of vulnerabilities.

At present, government and other organizations have few possibilities for requiring software manufacturers to safeguard cybersecurity in their products. As a consequence, problems arising from vulnerabilities largely come to lie with the user of a product. Users must therefore be particularly aware of this fact when purchasing software. Once purchased, users can do little more to check whether a product is safe.

Position of end users in relation to the manufacturer

Certain (large) users, such as government organizations and vital operators, are able to use advanced software and extensive analyses to search for vulnerabilities in software, for themselves. However, not all customers are in a position to test or reverse engineer the software for themselves, or to autonomously perform a full risk assessment (see also section 4.2 on information asymmetry and transparency). Interviews also reveal that not all organizations know how to lay down and enforce requirements and hold a manufacturer accountable. Manufacturers usually have agreements stipulate that they have limited liability for the consequences of any vulnerabilities in software. This makes vulnerability a problem for the user and not the manufacturer.

In addition, in the conditions they impose on the purchase and use of their software, manufacturers prohibit users from 'opening up' the product to see how it works, and to identify the components that make it up. This restriction is imposed by manufacturers on the basis of corporate confidentiality. These agreements form obstacles to organizations in subjecting the product to their own examination, and reporting vulnerabilities that are found during such an examination. Finally, via their terms and conditions, manufacturers specify that they cannot be held liable for the consequences of vulnerabilities in the software.[132]

Statutory requirements

Besides the imposition of requirements on a software product by the users, few other requirements are imposed by government for placing software on the market, maintenance during the lifecycle and the role of the manufacturer during incident management. The Wbni[133] Act requires providers of essential services to take security measures with respect to their network and information systems (e.g. reporting cybersecurity incidents), but this does not apply to software manufacturers. The above observation shows that in this system of parties, in particular with regard to legislation and regulations, there is a clear shortfall on the side of the manufacturers.

National initiatives

There are a series of initiatives aimed at arriving at legislation and regulations for the placing of software on the market. The Dutch ministry of Economic Affairs and Climate Policy and the ministry of Justice and Security, for example, have come up with an initiative in the form of the roadmap for Digital Hard- and Software Security (roadmap DVHS) in which they propose a package of measures aimed at preventing security problems in hardware and software, to detect vulnerabilities and to mitigate their consequences.[134] The measures in this roadmap are aimed both at prevention, detection and mitigation and include statutory requirements and the imposing of liability on manufacturers for damage suffered as a consequence of cyber insecurity. Concern for liability should serve as an incentive for manufacturers to take preventive measures or to limit damage. These measures are aimed specifically at smaller devices (IoT[135]), but are universally applicable to other types of software. The question that emerges is to what extent these measures should also be applied to safety-critical software and software in general.

International initiatives

Various international governments have taken the initiative to tackle the shortcomings in legislation and regulations. On 27 June 2019, the European Cybersecurity Act came into effect.[136] These new rules for cybersecurity among others reinforce the mandate of ENISA[137] and introduce a cybersecurity certification framework. Another recent example of an initiative in the field of legislation is the US cyber legislation, that imposes requirements on software purchased by government.[138] Australia also has plans for

132 Cyber Security Council (CSR), *Integrated approach to cyber resilience*, 2021; Tjong Tjin Tai, E. and Knoops, B., Duties of care and diligence against cybercrime (*Nederlands Juristenblad* 24-04-2015, volume 16), 2015; Anderson, R., *Security Engineering*, 2020.

133 Security of Network and Information Systems Act (Wbni) for digital service providers, see https://wetten.overheid.nl/BWBR0041515/2021-07-01

134 Ministry of Economic Affairs and Climate Policy and ministry of Justice and Security, *Roadmap for Digital Hard- and Software Security*, 2018.

135 Internet of Things, for example a smart TV, a smart refrigerator, connected temperature sensors, etc.

136 https://ecer.minbuza.nl/-/europese-cyber-security-act-van-kracht;
https://digital-strategy.ec.europa.eu/en/policies/cybersecurity-act

137 Originally the European Network and Information Security Agency, currently called the European Union Agency for Cybersecurity

138 https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/,
https://www.nytimes.com/2021/05/12/us/politics/biden-cybersecurity-executive-order.html

improving the regulation of cybersecurity.[139] The focus of this proposal is on IoT and organizations processing personal information. With regard to software safety, Australia is concentrating its efforts on stricter agreements on responsible disclosure as an incentive for manufacturers to accelerate the patching of vulnerabilities. In China, the exploitation of vulnerabilities is punishable by law, and sanctions are to be introduced for manufacturers that fail to release patches for reported vulnerabilities.[140]

In its latest report of recommendations, the Cyber Security Council (CSR) concluded that despite a number of important initiatives, both within the European Union and the Netherlands, there is still no comprehensive mechanism of responsibility for hardware and software security.[141] According to the CSR, manufacturers must be held more responsible for economic damage as a consequence of failing in their duty of care with regard to cybersecurity. This duty of care should help protect citizens and businesses against cybercrime.

Enforcement

If the enforcement of statutory requirements is implemented by means of certification of software, there remains a risk of perverse effects. The certification body after all has a business model in respect of the parties wishing to be certified, while for the certification of its software, a software manufacturer can opt for the route of least resistance. Competition between the different certification bodies does not always bring about improved standards and can in fact result in a race to the bottom (the principle of *maximum complacency,* whereby the manufacturer opts to have certification by a single certifying body confirmed, and objects to any attempt to encourage it to improve its product).[142]

139 Commonwealth of Australia, *Strengthening Australia's cyber security regulations and incentives,* 2021.
140 https://therecord.media/chinese-government-lays-out-new-vulnerability-disclosure-rules/.
141 CSR, *Integrale aanpak cyberweerbaarheid (Integrated approach to cyber resilience),* 2021.
142 Anderson, R., *Security Engineering,* 2020.

Past experience: Common criteria, ISO 27001 and BitSight

Common Criteria
The Common Criteria for Information Technology is an international standard for computer security. This standard faces a number of problems: certification costs are high, the standard is described in generic terms (the technology has been left out, including usability, an essential parameter for security), the standard is not capable of responding successfully to rapid developments in practice/application, there is no uniformity in the application of the standard (for example strict in Germany, very loosely defined in the Netherlands) and the standard includes no elements of liability.

ISO 27001 standard
The ISO 27001 standard[143] above all works for businesses as a means of earning money. Certification costs a great deal and is a source of income for the certification bodies. When a company applies for a certificate, the certification body is dependent on the information provided by the company. It is therefore possible for the applicant to indicate that certain security measures have been taken, while they have not actually been implemented in practice. There is no actual independent evaluation. Almost all major leaks have occurred in companies certified according to the 27001 standard.[144]

BitSight
Unlike the ISO 27001 standard, a private sector initiative, BitSight is a company that monitors the Internet in search of servers of companies and government institutions. Any server that is discovered is scanned and awarded a security score (for example on the basis of how many of its servers are (not) patched). As a consequence, BitSight is not dependent on information provided by companies (the applicants in ISO 27001 certification) and arrives at a score, on the basis of its own scans. However, this too has negative effects. For example, companies are cautious in deliberately linking vulnerable servers to the Internet (for example for training employees, students, etc.). As soon as servers of this kind are observed by BitSight, this has a negative influence on the company's security score.

Enforcement is only possible if manufacturers are required to be transparent about how their software works, in such a way that third parties are able to assess its safety. The Executive Order on the improvement of cybersecurity in the US focuses on this point and identifies an urgent need for stricter and more predictable mechanisms for ensuring that products function more safely, in accordance with their intended purpose.[145]

143 An ISO standard for information security. See https://www.iso.org/isoiec-27001-information-security.html
144 Anderson, R., *Security Engineering*, 2020.
145 https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Economic incentives

The examples in this investigation reveal that software products are dynamic. This is because they are regularly updated for the addition of new functionalities and for repairing vulnerabilities. At the same time, these products often have a long history, as they can be built on existing components. This can make it a costly investment for manufacturers to tackle the root causes of any insecurity, as described in section 4.1.1. Tackling root causes would require them to rebuild software that is the result of decades of development.

There are few economic incentives to compensate for this investment. Insurers not only insure organizations that use software but also the manufacturers that make the software. In this latter role, the insurers demanded of the manufacturers that they pass on liability for the consequences of unsafe software to the actual organizations that use the software. The Cyber Security Council writes that insurers ideally impose requirements on both the manufacturer and the organization using software.[146]

A manufacturer can also experience an economic incentive if the value of its shares falls as a result of an insecure system (shareholders). Shareholders of SolarWinds, for example, sued the company: according to the shareholders, the private equity companies that own SolarWinds sacrificed cybersecurity in favour of short-term profit ('goldrush among investors in SaaS business').[147]

In addition, the material obligation to remove software insecurity can deliver an economic incentive for a manufacturer to do more to prevent software with vulnerabilities being placed on the market. At present, this economic incentive lies exclusively with the users of the software.

146 CSR, *Integrale aanpak cyberweerbaarheid (Integrated approach to cyber resilience)*, 2021.
147 https://www.scmagazine.com/home/solarwinds-hack/solarwinds-lawsuit-claims-private-equity-owners-sacrificed-cybersecurity-to-boost-short-term-profits/

Tracing and recall in the food sector[148]
In the food sector, food companies are required to be able to trace to whom they have supplied their food products. This obligation applies throughout the food chain, from primary production (such as agriculture, livestock production and fishery) through to the consumer who eventually eats the food. In every link of the chain, a food company must be able to trace where the raw materials came from, and to whom they have supplied their products. This obligation is known as traceability. If a food company discovers that it has placed unsafe food on the market, within four hours it must be able to compile a distribution list with all buyers[149] and purchased products, which on request is submitted to the authorities.

Food companies are also required to recall the unsafe foods on their own initiative, or if so instructed by the authorities. In practice, it is sufficient for the authorities if a food company restricts itself to a publication in a daily newspaper and/or on its own website, but an 'absolute recall' means that the food company must warn its customers as directly as possible, and call for them to return the products, possibly even collecting the products itself, from the end user. This latter action is for example carried out for recalls of passenger cars if the safety problem is so serious that the car may no longer be used on public roads.

Regulation and liability also play a role in the occurrence of vulnerabilities. At present, governments and other organizations have few possibilities for obliging manufacturers to safeguard cybersecurity in their products. Users do not always know how to impose requirements, and force manufacturers to show accountability. This makes vulnerability a problem for the user and not the manufacturer.

There are practically no rules for placing software on the market. The current free market for software products imposes almost no requirements on the sound management of security risks. Identifying vulnerabilities is a time-consuming task, that demands much manpower and as a consequence is costly. In certain cases it can be necessary to completely rebuild a product in order to tackle the underlying (safety) problem. The absence of economic incentives explains why manufacturers at present do not consider this option.

148 Based on the idea that there is a chain from producer to consumer via a number of intermediate steps, the compulsory traceability for every company in the food sector applies one step back and one step forward in the chain (excluding the step to the end user or consumer). Source: Article 18(1) of Regulation (EC) no. 178/2002 in: Guidelines for the enforcement of Articles 11, 12, 14, 17, 18, 19 and 20 of (EC) Regulation no. 178/2002 laying down the general principles and requirements of food law (26 January 2010).
149 For the last link (the end user or consumer), the tracing obligation does not apply, but certain retailers do record (some) deliveries to consumers (online orders, customer loyalty cards, etc.).