WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Seminar Paper

Security Concerns in Proprietary and Open Source Software

by

Fabiola Welzenbach, h1552712

handed in on the 03.06.2020

Supervisor

Univ. Prof. Mag. Dr. Rony G. Flatscher

LV Seminar aus BIS, 4167

Erklärung:

Ich versichere:
dass ich die Seminararbeit selbstständig verfasst, andere als die angegebenen Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

dass ich dieses Seminararbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/ einem Beurteiler) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum 03.06.2020 Fabiola Welzenbach

[illegible]

Abstract

In times of digital transformation and increasing amounts of cyber-attacks, seeking control over confidential information for companies is on top priority. Thus, this work highlights different aspects of considering open source and proprietary software of security concerns. The main research question is whether open source or proprietary software is more secure. To answer this question, a basic guideline of open source and proprietary software gives a first understanding of software in general. Then, advantages and disadvantages have been evaluated of both types and subsequently, typical attacks, real case attacks and strategies to overcome security concerns are involved. The goal of this paper is to help companies to understand the down- and upsides of both software types according to security concerns, to then decide, which software will fit to their IT strategy.

Table of Contents

List of abbreviations

FSF	Free Software Foundation
OSI	Open Source Initiative
OSS	Open Source Software
VW	Volkswagen AG
CMS	Content Management System
NVD	National Vulnerability Database
SCAP	Security Content Automation Protocol
CVE	Common Vulnerabilities and Exposures
SCA	Software Composition Analysis
MSRC	Microsoft Security Response Center

1 Introduction

In the last years the term “digitalization” has become a major concern for all companies around the world. With digitalization the world is changing faster than ever before. New products have been developed, small businesses start becoming challenging competitors and new business models were established. To stay competitive companies, have to adjust their systems, their value chains and their whole business strategy. But after the process of digital transformation has been successfully implemented, it entails great advantages. Due to Industry 4.0, robots can now perform difficult tasks that humans could not do, and sensors can be used to provide accurate data and thus response times can be shortened. Digitalization plays a great impact and only these two examples out of many more, show how valuable digitalization has become. But as digitalization cannot only be used in a positive way, it gives attackers an increasing opportunity to destroy and steal valuable and confidential data from companies and their customers. Due to the Corona Crisis this has become especially intense as with everyone being in home office, attacks have become easier. This can be explained as the network of companies is normally much more secure than it is at the employees’ home. Overall, this shows that security is a major concern of today’s businesses in regard to digitalization. To ensure security, businesses have built up teams in their IT department that focus on security and to reduce the risk of such attacks. These departments are of particular importance if the company is a major enterprise and a failure would have far-reaching consequences. In the worst case, for example, if there would be a successful attack on an airport, the entire air traffic would be stopped, passengers would miss their connecting flight and the cost related to this would be enormously high. Thus, security should be handled as an important asset in a company. As these attacks are most likely done on software, this paper gives an overview on security concerns of open source software (OSS) and proprietary software. The overall question that runs like a red thread through the paper is, if whether open source or proprietary software is more secure or not. To answer this question, the first part of this paper gives a basic understanding of software in general and open source and proprietary software. After that, the focus lies on the different licenses, which also play an important role in terms of security. Subsequently, general advantages and disadvantages of open source and proprietary software are analyzed.

Based on this, security concerns are then specifically assessed and real examples highlight the issues. After considering the issues that come along with open source and proprietary software, security strategies to overcome security concerns are mentioned. Furthermore, the change from the proprietary software giant Microsoft to an open source software contributor has been analyzed and security suggestions for Microsoft users are presented. Finally, the initial question of whether proprietary software or open source software is more secure is answered and the results of the work are briefly summarized

1.1 Definition

This chapter gives first a brief overview of the definition of software in general. Subsequently, open source and proprietary software will be explained and at the end of this chapter the focus lies on the different licensing types of both software types.

1.1.1 Definition of Software

To understand the concept of software it is first of importance to have a basic understanding of hardware because software is built up on hardware. Basically, the term hardware implicates all physical units, which built up a computer system. This could be a screen, keyboard or a system unit. To be able to use the hardware, suitable software is necessary. Thus, the programs available on computer systems are called software. The executable code, which controls computer behavior and operations, can be defined as software and is used to describe a broad range of applications, programming languages and procedures. Software controls, integrate and manages individual hardware components of a computer system, to enable users not having to get into detail with low-level details of a computational system (Bouras, Kokkinos, & Tseliou, 2012).

Overall, software cannot be seen as an inseparable whole. Because a complex software system is made up of multiple interacting software components, formed by several layers of components. Thereby components depend on other components and/or contain further subcomponents themselves. Software components can be categorized as infrastructure, industry, application and development components

(Hansen, Mendling, & Neumann, 2015). A distinction is therefore made according to the intended use:

- The **system software** is necessary to control the existing hardware. The central component of the system software is known as the operating system, which has multiple tasks. The main task is to control the computer, directs instructions and data to the processor, organizes and manages internal and external memories and connects devices. It is also the interface between the computer and the user. The operating system evaluates the commands entered and the execution of these instructions is analyzed. As the hardware of a computer system exists of one or more processors, memory, network interfaces and other devices, it is of importance to design programs that manage the complex interaction of these components and use them in the right way (Wiedemann, Holey, & Wiedemann, 2007).

- Second, in general the **infrastructure software** provides the technical infrastructure for other components, which usually offers the user only a small added value when it's on its own because it is essential for the function of the overall system. Important to consider is that the infrastructure software makes the development of application software much easier. Infrastructure components include, database systems and graphical user interfaces (Hansen, Mendling, & Neumann, 2015).

- The **application software** is used to enable the computer to be used for a wide range of applications. The utilization of suitable application software allows the user to design and edit texts, images or tables with the computer. Also, to manage data stocks in databases or to access information on the Internet (Wiedemann, Holey, & Wiedemann, 2007).

- Based on the operating system, the **development software** allows the programming of almost every program. An important part of a development software is the programming language. The development software is needed to create system, application and development software (Hansen, Mendling, & Neumann, 2015).

Based on the overview about the definition of software and its components, this leads to the understanding of the main differentiation of software – proprietary and open source software.

1.1.2 Definition of Proprietary Software

The adjective “proprietary” comes from the Latin word *proprietas* “owner”. Meaning “held in private ownership” (Online Etymology dictionary, n.d.). Based on this general understanding of the term proprietary the following paragraph gives a short overview about proprietary in connection with software. Proprietary software is software with limitations on using and copying it, normally by a proprietor. To prevent modification or copying software this can be achieved through technical or legal means. Legally, through software licensing, patent law and copyright and technically through releasing machine-readable binaries only and withholding the human-readable source code. The term proprietary software describes software, which is not free or semi-free software used by the Free Software Foundation (FSF). Primarily, proprietary software has an owner, who has control over the software and its source code. According to proprietary software the owner is of major importance, however to "free software" the user plays a major role. Windows OS and Mac OS are the most well-known examples of proprietary software (Sahoo & Sahoo, 2016).

1.1.3 Definition of Open Source Software

Most software programs downloaded or purchased are only available in the compiled ready-to-run version. This means for developers; it is enormously difficult to change the version and to have a deeper understanding on how the developer has developed different parts of the software. However, many companies see this as a positive effect as it avoids competing companies to copy their code and using it in a product. On the other hand, by using open source it basically describes the opposite. Its source code is open to everyone and adaption as well as modification is suggested. A goal of open source software is that the application will be more useful, error-free and gives everyone, who has knowledge in programming the opportunity to adjust the source code. (Sahoo & Sahoo, 2016) In order to share open source software ten criteria by the internationally accepted open source definition must be met. Therefore, only licensed software under an Open Source Initiative (OSI) accepted open source license can be called open source software. The ten following criteria for distributing open source software are as follows and can be found on https://opensource.org/osd:

1. **Free Redistribution:** The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
2. **Source Code:** The program must include source code and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.
3. **Derived Works:** The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software
4. **Integrity of The Author's Source Code:** The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
5. **No Discrimination Against Persons or Groups:** The license must not discriminate against any person or group of persons.
6. **No Discrimination Against Fields of Endeavor:** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
7. **Distribution of License:** The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. **License Must Not Be Specific to a Product:** The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.
9. **License Must Not Restrict Other Software:** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.
10. **License Must Be Technology-Neutral:** No provision of the license may be predicated on any individual technology or style of interface.

Considering all criteria together it can be seen that when using open source software, it comes with restrictions. When talking about open source software many people use the term free software. Even though these two terms are related, they have a slightly different meaning. The definition of the concept of open source was developed by the Open Source Initiative. Historically the term “open source” was invented at a strategy meeting in 1998, in Palo Alto and was suggested by Christine Peterson. This announcement took place right after the release of the Netscape source code, which represented a great chance to promote open source development. Eric Raymond and Michael Tiemann, who both attended at the strategy meeting in Palo Alto were later the Presidents of OSI. Due to the support from key individuals like the founders of Apache, Python and Linux the term “open source” was very well and quickly accepted. The original purpose of OSI was to build an educational organization with the focus on declaring and protecting the "open source" label.

On the other side, the term “free software” was invented by the Free Software Foundation to protect and support free software, founded in 1985. Even though they shared the same goal to support the distribution and development of free software, the OSI had disagreements with how FSF promoted the free software. The OSI had the opinion that free software was a functional rather than an ideological issue. But this was not the only issue the OSI disagreed with. Another one was that the OSI was concerned about the word “free”. Because “free” software can mean the software does not cost anything, but the source code is still not available. For example. Microsoft Internet Explorer is free of charge, but you still cannot see the source code, even though its “free”. Also, some software is labeled “free” but the source code is covered by a license agreement and copyright with further restrictions (Why `Free Software' Is Too Ambiguous, n.d.). These examples highlight the issue that free software is not always “free”. On the other side, the term “open” was used to describe the "absence of hardware vendor lock-in" in the computer industry back in the 1980s. Later the term was defined by the OSI that “open” applies to the open software source code, which is visible, accessible and free to change the code without paying additional fees. As both terms are related to each other choosing between the two related terms “free” and “open” can lead to confusion and is the reason why today some people use both or the term "free and open source software" (FLOSS) (Open Source Initiative, n.d.). In this paper the focus lies on open source software and free software is only sometimes touched on.

2 Licensing

In this chapter the focus lies the different types of licensing of open source software and proprietary software. The differences of each types will be explicitly highlighted and examined.

2.1 Licencing – Open Source Software

Even though the original purpose of open source was to build up an educational organization with an open source code that can be modified and shared without any hurdles to overcome, the business in which open source software is now used has changed in a different way.

As stated on Github.com over 50 million developers worldwide are currently registered on GitHub. Overall, GitHub is the world’s largest open source community (Github, n.d.) with also contributors from big tech companies. For example, the five most active employees are contributors from Microsoft, Google, Red Hat, Intel and Facebook (Brodsky, 2018). This large knowledge of the community combined with also skilled employees from big tech companies gives enough incentives to build up ideas for businesses. Thus, new business models were established, and companies started to copy the source code of well-known projects, made some changes and gave the software a new name and offered this “new” software at the end to customers as proprietary software. The most common companies, who used the open source community to build their business on are most likely cloud providers (Honsel, 2020). As this was not the idea of open source initially, open source activists started to make use of licensing open source software to restrict business to grow from making use of the open source community. Richard Stallman, the founder, of the Free Software Foundation and one of the most popular licences (GPL) once said “*My work on free software is motivated by an idealistic goal: spreading freedom and cooperation. I want to encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better.”* (Stallman, 2002). This statement and the problem of making use in the wrong way of open source software emphasizes the importance of licenses.

Open source licensing declares that software can only be used in commercial applications under certain conditions and is a legally valid and binding contract between the developer and the user. As long as developers use open source applied to specific conditions defined in the license, developers that make components into open source can use this software. Over 200 open source licenses exist and each states the terms and conditions what users are allowed to do with the software components. Based on the conditions and restrictions, two main categories such as copyleft and permissive can be found. As copyleft is not the contrast of copyright, it is first important to understand the term copyright. The term copyright refers to the law that work is restricted to be shared without the permission of the copyright holder whereas the copyleft license states that user have the permission to use, modify and share it. However, the user has the

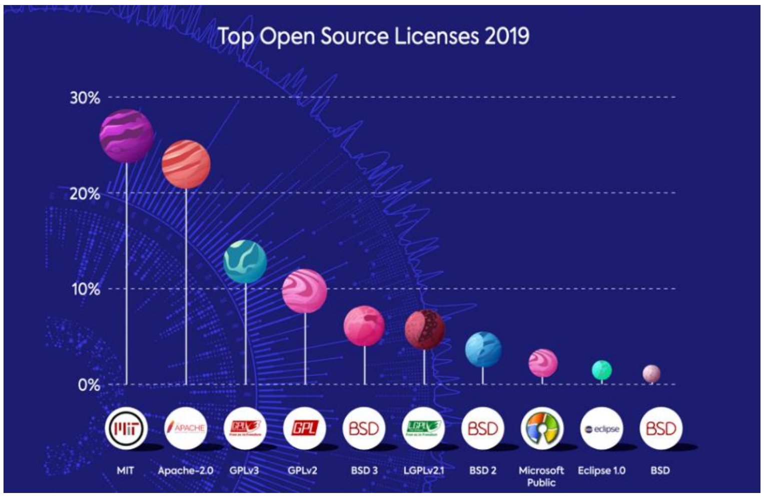

Figure 1: (WhiteSource, 2020)

obligation to publish modified program code again under the same license, meaning that programs under copyleft licenses may neither be combined with proprietary software nor turned into such. In a nutshell, the user has to make the open source code for other users open as well. In contrast, permissive open source license is related to the copyleft open source license but has a major difference. Because similar to copyleft it guarantees to use and modify the source code. However, the crucial difference lies in the distribution of the source code, as the permissive license also allows to distribute proprietary derivative works. Currently, as shown in Figure 1, the top three open source licenses are MIT, Apache 2.0, General Public License version 3 (GPLv3), General Public License version 2 (GPLv2) and BSD3 (WhiteSource, 2020). The most widespread MIT permissive license was founded by the Massachusetts Institute of Technology in 1988 with no obligation to disclose the code and is used by: Bitcoin, X11 and Ruby on Rails. Secondly, the Apache 2.0 license is as well as the MIT license a permissive license and was founded in 2004. One main principle is that changes in source code must be identified. Most commonly used by Android, Apache webserver and Open Office. The third most widespread license is the GPLv3. This is in contrast to the first two licenses a copyleft license, which was introduced in 2007. A major characteristic of the GPLv3 license is the internationally orientation and the compatibility with third party licenses - used by Linux (Honsel, 2020). Hence, by considering all the different licenses it can be seen that the complexity of open source software in the last years has changed intensively. Open source software and its different licensing types have become a major impact when using open source software.

2.2 Licensing – Proprietary Software

Compared to open source licensing, proprietary software licensing varies. Thus, different companies make use of the way licensing can be used with proprietary software.

It is often stated that developing the first software costs millions and the following copies cost only a bit. But this statement is actually a cliché because packaged goods can be developed for a few dollars each. Therefore, many proprietary software providers take advantage of that and sell continually high-margin licenses of the same

software. By improving these versions with new licenses, gives then vendors the possibility to request extra fees. As this is not enough, proprietary software providers see another business opportunity by using this concept, because they also sell their support and maintenance services for that software. This whole business concept only works if the source code is proprietary (Landy & Mastrobattista, 2008).
Also, entire businesses have started to “rent” the source code. This means that proprietary providers impose, over a contractual agreement, on each license to hold the source code secret and to distribute only binary derivatives of the source code. Especially interesting is this offer for licenses, who need to modify and adapt the source code (Landy & Mastrobattista, 2008). One of the most popular license agreements is the End-User License Agreement, short EULA. This license agreement is mostly used as a contractual agreement by software developers and users of the software. For example, Mac, Android or Windows. Basically, the license can be seen as guideline for terms and conditions and can set rules in which way the application can be used (EULA, n.d.). The Microsoft EULA’s include the following note. “You may not copy or post any templates available through Internet-based services on any network computer or broadcast it in any media.” This means that if a publisher would use a Microsoft Word template, and would adapt it to his business, the publisher would be in violation with Microsoft’s EULA (DiBona & Cooper, 2005).

3 Advantages and Disadvantages

This chapter includes the advantages and disadvantages of open source software and proprietary software. The aim is to give a basic understanding of general concerns. Especially security concerns, will be highlighted in section 4.

3.1 Advantages and Disadvantages of Open Source Software

According to a market survey of bitkom, more than 800 companies with more than 100 employees in Germany have been asked to give an overview of the extent to which open source software is already used in Germany and to what extent companies are participating in using and developing open source. The outcome of this survey has been summarized in the "Open Source Monitor 2019" of bitkom. One part of this survey was to examine the advantages and disadvantages these companies experience with open source software. Overall, it has been examined that nine out of ten companies with 100 or more employees see advantages in using open source software for their company.

About 17 % of the companies named the cost savings factor as the greatest benefit, as mostly no fees have to be paid for licenses, which adds another value. In regard to the security factor, in total twelve percent of the companies have named security-relevant advantages. Nine percent appreciates the high level of security provided by regularly updates, and the remaining three percent appreciate low error-rates and high stability (bitkom, 2019). This positive effect occurs because the source code is open and “holes” are easier to be found and are therefore more secure and reliable (Landy & Mastrobattista, 2008). According the openness factor, 9% see the advantages in being independent from proprietary software providers. This aspect will be explicitly mentioned in the part of the disadvantages of proprietary software. The uncomplicated implementation of individual software adaptations, the access to the source code and the large selection of open source components have been rated with 7 % each as an advantage. Also, every twentieth company (6 %) appreciates the support of open standards and interoperability, the diversity of open source providers (5 %) and the compatibility between components and tools (5 %) as the greatest benefit of open source software. With regard to the cooperation and innovation factor 8 % value the exchange of knowledge within the open source community (5 %) and the promotion of innovation and competition (3 %) as benefits in the use of open source software (bitkom, 2019).
As we can see this survey has shown that most companies see many different advantages in using open source software for their business. Obviously, the main advantage when using open source software are the cost savings. This most likely results as when using open source, normally no or less license fees have to be paid. Also, companies like the advantage of the fast implementation. This may result in the reason that open source is always available and easily to access with highly skilled employees. Moreover, the benefit of the community has also been mentioned. Most likely this advantage comes from the motivation of the people, who are using open source software and the history of freedom and cooperation. Last but not least, it is also important to note that 9 % value the benefits of security. Thus, it shows how

necessary security concerns are for companies. More deeper insights will be explicitly explained in section 4.

However, the advantages are also offset by various disadvantages. In this paper these disadvantages will be divided into four categories: personnel, insecurity, IT security and supply. The biggest problem these companies have mentioned was a lack of skilled workers. Because experts in the company, who could, for example, adapt and further develop the software to individual needs are rare. The reason is that only a half percent of the world's population knows how to code (McGovern, 2019). In this context, a lack of training opportunities (6 %) and familiarization effort (5 %) for the specialists are seen as disadvantage, which also could lead to the critique that community-supported products will not be repaired (Landy & Mastrobattista, 2008). In regard to the insecurity context, 6 % see unclear warranty situation, the uncertainty of the future of open source software, no or imprecise supplier liability and legal uncertainties regarding licensing as disadvantages. While 12 % of the companies cite security-related aspects as advantages of open source software, a total of 7 % tend to see these as cons. 4 % complained about security gaps and other 3 % saw the susceptibility to errors as a disadvantage. These aspects of insecurity and IT security will be focus in the section of security concerns more deeply. The last section, which was examined are the disadvantages of supply because not every company appreciates the large selection of open source providers and components as an advantage. Thus, 6 % see this as a disadvantage. A total of 6 % sees a problem in the lack of OSS solutions for their specific use cases and the conversion of proprietary software to open source software as very complex, time-consuming and costly. As the development is therefore very uncertain, another 6 % rate the warranty situation for open source software as unclear. Further companies criticized the supplier liability (3%) and 2 % cited legal uncertainties regarding licensing. The comparison shows the advantages outweigh the disadvantages (bitkom, 2019). Overall, as the main lack of open source software by this survey is the knowledge of skilled workers, it can be seen that this is a worldwide problem and therefore, trainings and education is much needed to further promote open source software.

3.2 Advantages and Disadvantages of Proprietary Software

After the advantages and disadvantages of open source software have been analyzed it is now important to compare the results with proprietary software. Thus, this chapter will set the focus on proprietary software.

One of the main advantages of proprietary software for the end user is the ease of use – the usability. As proprietary software is focused to serve directly the end user, it is developed with a smaller scope and fewer functions. The importance of usability is that not only developers but also application users can use the software. Furthermore, proprietary software providers have to check and maintain their software products regularly to be successful. Because the software they offer is designed for a long and prosperous future with many paid upgrades. This leads to the positive result that proprietary products are stable over time. Another positive aspect, which may sound first irritating is the advantage of ownership. Typically, a company acquiring proprietary software gets the full rights to their software product. The price is high, but the customer support package comes normally with support and supply with updates, reworked documentation and error corrections. Also, most of the time, these packages are developed according to the needs of the companies. This gives the advantage that companies appreciate the customized support of proprietary software provider. As the scope is compared to open source with fewer functions etc., training and customer care is more comprehensive, accessible and concise, especially when it comes to the most important first step, the integration (Optimus Information, 2015). As also with proprietary software not only advantages appear but also disadvantages, the next paragraph focuses on the disadvantages.

A well discussed topic in times, when unknown incidents like the Corona virus occur, dependencies may cause a great debate. One learning process was definitely the dependency of medical equipment from Asian countries. Due to globalization and David Ricardo’s theory of Comparative Cost Advantage (1817) that each country of the world should focus on the export of their resources and strength with less costs, countries have become more and more dependent. The current crisis has shown the impacts of this enormous trend between dependency and globalization worldwide.

This dependency does not only appear when it comes to raw materials and medical equipment but also with systems. Because today companies like SAP, Microsoft and Apple have gained so much power that many companies have become dependent. The problem is most companies use the applications of individual manufacturers uniformly because the service these big cooperates offer, is so attractive that looking for alternatives does not make any sense. Also, individual manufacturers ensure that the various updates are robust and that the products remain compatible. The type of administrative management or the arrangement of functions remain the same over years and the employees do not need any additional training and can carry out their work. However, by conducting business based on one system, dependency takes place, known under the term "vendor lock-in". The dependency has grown so far that switching to another product is so difficult, due to costs or trainings, and may even be impossible. In the field of software, a possible vendor lock-in is caused by proprietary file formats that can either not be opened and generated by competing applications or only incorrectly. For example, proprietary protocols, such as those used in the network environment, also enable data to be exchanged exclusively between the programs of one manufacturer. Furthermore, employees who, after years of getting used to one product, are unwilling to adapt themselves with another programme. Also, switching costs complicate the changeover, because if the costs of switching to competing software are likely to be significantly higher than the license costs for the product currently in use, no change will take place. But what has bad effects on the one hand, can be good on the other hand. Meaning that operating a proprietary system may also be an advantage for companies. However, in order to resist dependency, companies and states strive for diversification and cooperation with more parties to reduce the risk of dependency. Therefore, open source software came into the spotlight recently. As a competitor to proprietary software, open source represents a new way of developing but has also big obstacle to overtake in business. Because being dependent is just one issue, which occurs when considering proprietary and open source software (Freist, 2020).

Obviously, another big downside is that users are not able to modify the source code as by definition, the internals of closed source software are blocked for viewing. The users are only able to report errors that have occurred and wait for correction, if there is no workaround or patch. This consultancy process is not only time consuming but usually more challenging for customers to make adjustments or optimizations in their end product (Optimus Information, 2015). Considering the fact of being dependent is a huge downside of proprietary software. Hence, companies should seek for alternatives and also always reconsider the trust in one company when using proprietary software.

4 Security Concerns of Software

In general, IT-security aims to protect information systems from unauthorised access and unauthorised use, by guaranteeing accessibility of information, reliability and integrity. According to software security this means that software has to still function, even though it has been attacked. Most typical software vulnerabilities are cross-site scripting and SQL injections. Problems at the architecture or conceptual level are security holes and concerns at the implementing level are security failures, which are even more difficult to detect (Sametinger, 2013). Thus, researches like Mouelhi, Kateb and Traron have found out that security tests have become more important than ever before in the last few decades. Solutions according to security modelling and implementations of security mechanisms have been offered by many scientists. To prevent security loopholes in the system, testing of implementing security mechanisms is from major importance. One example of implementing a security mechanism is the access control, which is very important as it secures that only authorized users can get access to secured resources in a certain system (Mouelhi, El Kateb, & Le Traon, 2015). Even though possibilities to protect a system exists, unfortunately, most applications today lack security and one of the weakest parts is software security. Companies, who use either proprietary or open source software have to deal with many risks.
Thus, companies have to modify their application development lifecycle to prevent hackers to intrude, which requires a strong, internally implemented software security (Mehta). For a better understanding the focus of this chapter is to consider security concerns of proprietary and open source software individually and then evaluating possible attacks companies have to deal with.

4.1 Security Concerns – Open Source Software

Before diving deeper in the main topic of security concerns of open source software of this work, the following paragraph gives a short overview of the adjustment and use of open source in companies. Subsequently, security concerns of OSS will be discussed in direct comparison with security concerns of proprietary software.

Getting back to the analysis of bitkom shows that the majority of the companies evaluated, have positive attitudes towards OSS (75 %). Only 4 % of the companies are more the less against OSS and have critical thoughts about it. Also positive is that already 69 % make actually use of OSS. However, most of the companies do not change the open source code. In total this shows that there is an existing interest, companies use it to some extend but there is still room for improvement. As some disadvantages of OSS, were already discussed, one reason why still 27 % do not use OSS is the fear of security concerns that come with OSS (bitkom, 2019).

When it comes to the question whether open source software or proprietary software is more secure the opinions differ drastically, because some people have claimed that open source software is more secure than proprietary software and others do not agree. At first sight, because of the open source code, it seems obvious the claim that proprietary software is more secure than OSS is right. Because with a published source code this means hackers can easily review the source code. However, a typical claim of OSS proponent is the argument, that as the source code is published to everyone, which means everybody can review it, not only developers but also hackers. Programmers will detect vulnerabilities and report them and make OSS more secure (Payne, 2002). Of course, in these claims there is some truth in it, but it takes more aspects to consider whether open source or proprietary software is more secure. Due to the research taken, many papers state (e.g. Crispin Cowan: "*neither case is absolutely true: they are essentially flip sides of the same coin*") there is no right or wrong answer to this question. Thus, one of the main aspects of this paper is to give a comprehensive understanding if these statements are correct or false. In general, OSS gives attackers and defenders a great analytical possibility to find solutions against software vulnerabilities. The importance is, if the defender does nothing against security, then the attacker has a major advantage with OSS. But if the defender is willing to support security when using OSS, he can also profit, as he has access to security techniques that are normally not feasible with closed-source software. Because when using proprietary software, the user has only the option to accept the degree of security the vendor offers, whereas with OSS the user can chose how high the security standards should be (Cohan, 2003).

Before diving deeper in different aspects of security concerns of OSS it is necessary to first define, what is actually meant with security concerns of open source software. First of all, in this paper it is defined as either positive aspects or negative aspects of security concerns compared to proprietary software. Furthermore, it is partly defined between the security of the system and the risk connected with using the system. The question whether the system is secure or not by using OSS depends on the degree of security connected with the risk when using it defined by the combination of the probability of an attack and the associated damage.

According to the argument of proponents that OSS is more secure because many developers have access to the code and fix problems, a typical example of the strengths of peer reviewing will further discuss this claim. It is known that in terms of security the alleged difficulty for an attacker is to install a “backdoor” into an open source program. In general a “backdoor“ is malicious code, which allows to simply and secretly bypass security mechanisms by an attacker. This “backdoor” works if the source code is connected in some way to a legitimate program or system or is either embedded into a legitimate program or system. Thus, a “backdoor” in an operating system could allow a person to login from a specific IP address as a system administrator without using a password. Proponents of OSS argue that this “backdoor” is nearly impossible to embed secretly as many developers have also access to the source code and can reveal it.
For example, highlighting the success of detecting “backdoors” in open source software was the “backdoor” in Borland/Inprise’s Inter-base database software in 1992. When authorized programmers of the company embedded by mistake malicious code in a proprietary software. If hackers would have been informed about this “backdoor”, it would have been easy for them to have access to all installations of the database system. This could have mean for the company a very malignant outcome, which has not come true as the company decided to change the product to open source. Thus, after nine years having a backdoor in the system, the change to open source software

has helped to detect the mistake. This gives proponents of OSS a proof to claim that when using proprietary software, a malicious code will longer be undetected as with OSS.

Another negative example of backdoors in OSS is an error known as *Heartbleed* in OpenSSL. Since OpenSSL is an important service for encrypted communication, a bug in such systems is a problem that can affect millions of people simultaneously. *Heartbeat* is an extension of OpenSSL and should periodically check if the connection between server and client is still available. However, a programming error resulted in an attacker being able to retrieve 64 kByte from the client's main memory for each of these queries or for each response to this query. This gap remained undiscovered for 27 months and was commented by security specialist Bruce Schneider with the words: "On a scale of 1 to 10, this error is an 11". Any information could be read from the memory, including passwords in plain text. The vulnerability was caused by a thesis of two students, who extended the *Heartbeat* code and made a programming error. This bug remained undiscovered even after a core developer of OpenSSL had checked it and continued to spread. This has end up in far-reaching consequences for encrypted communication, as the security of the protected connection between computer systems was neglected. A discussion about the quality of Open Source soon followed. In the end, the mistake was due to the fact that the OpenSSL Foundation was completely understaffed at that time and could only afford one full-time developer. The vulnerability was simply overlooked. In the meantime, the OpenSSL Foundation has reorganized itself and programmers from Oracle and Akamai are taking care of the development. Sponsors from the business world also support the project financially. Hence, it can be seen that when using open source software, it should be deeply tested to detect vulnerabilities and to prevent bugs from spreading. This is especially from importance in accordance to encryption issues (Borchers, 2020).

Furthermore, a major advantage for open source users are the extra rights for the usage of OSS, which are not given with proprietary software. Companies can decide whether to make a security audit or not. For corporates with an extreme interest of high security standards the possibility to make internal security auditing is a great advantage and most of the time obligatory. But not only auditing security issues of the source code is an advantage, also the possibility to adapt and modify the source code is beneficial. If a company seeks high security standards, the ability to modify this software to specifically meet the requirements is even more from importance (Payne, 2002). Hence, if a company has skilled IT specialists, the possibilities, which can be used when using open source software are greater than with proprietary software.

Even though OSS is used and implemented in many systems, there are also downsides to consider. Probably the most well-known fact that the source code is open gives also attackers the possibility to easily access the software. This allows the attacker to have access to a broad range of information, to seek for vulnerabilities and failures in order to be able to attack in a targeted manner. This also gives the attacker a major advantage as the attacker only needs to find one vulnerability to threaten the system, whereas the defender has to find all possible weaknesses, to protect it.
As the main principle of OSS is the openness of the source code, it means that systems with a large developer basis check and improve the code to a large extend and even though many developers check and improve the source code, it does not necessarily mean they identify all the bugs, which could compromise system security. This issue is highlighted by the statement of NFR Security's Ranum, "In my experience, not many people actually read through the code. They just look at the readme files. When I wrote the first public firewall toolkit, there were probably 2,000 sites using it to some degree, but only about 10 people gave me any feedback or patches. So, I am way unimpressed with the concept of open source." Thus, proponents of proprietary software are using this assumption to put the quality of closed software in spotlight. They argue that employees, who make money with proprietary software are doing their job way much better than the open source community (Lawton, 2002). In fact, this failure in identifying bugs often occurs, as open source developers are mainly interested in the progress of the development and further improvements and are not likely willing – as hackers - to invest time and energy on software auditing. Comparing this security concern with proprietary software, it shows that closed software is in this case is more secure as vendors pay their developers to carry out auditing (Schneider, 2000).

To find a solution to this social problem of not putting enough focus on auditing the source code, the Sardonix project was created. The aim of this project is to encourage the open source community to a higher security standard of the open source code by reporting who and which source code has been audited by a ranking system. It measures the quality by the amount of audited codes and the missed vulnerabilities, which have been detected by others. This gives auditors the chance to use the rank for their resumes. To audit the source code in an efficient way, static and dynamic analyzing tools exist. In the case of the static analysis tool, analyses about mistrustful code sequences, which could be vulnerable are made. On the other hand, as many security issues are undecidable when using the static analysis, the dynamic tool often helps. By using test loads while running the program, the developer can examine what the program does (Cohan, 2003).

Important to consider is also the fact when it comes to auditing security of open source that the quality of the software depends on the skills of the programmer. Normally for many open source projects every help is highly welcomed, and a selection procedure of developers does not exist, thus a quality check of the projects is missing (Bart, 2018).

Another aspect when it comes to security of open source and proprietary software is, that for example the US government requires also for IT products to pass a Federal Information Processing Standard audit before US agencies can buy them. This certificate issued by the government for using products by US agencies can boost the product. But as the costs for the compatibility testing is so high, mainly proprietary vendors are willing to pay it (Lawton, 2002).
Also, a further pain point of OSS is that when a company buys proprietary software, services and updates are also included. This means a software vendor is obligated to publish all vulnerabilities along with fixes and software patches to the user. This is different to the OSS community as there is no existing obligation to publish vulnerabilities. Hence, the company using OSS is highly depended on the information from previous OSS communities and security forums (Zhang, Malhotra, & Chen, 2018). Another problem shows the trend of focusing more and more on features whether than on the ease of use. Thus, highly developed system architecture has tended to be the focus, which leads that the applications are difficult to provide and to manage. By adding more and more software architectures, software vendors see this as a business opportunity. Because customers are retained for a longer period of time, as new information technology and hardware will only be implemented step by step and not at

once. The focus has been set in a wrong way, while security functionality was mainly left out (Schneider, 2000).

4.2 Security Concerns - WordPress

WordPress is nowadays the most popular open source content management system (CMS) in the world. More than 409 million people view over 20 billion pages each month and the users produce monthly around 70 million new posts and 77 million new comments (WordPress, 2020). In 2019 over 60 million people have chosen WordPress for their website design. Thus, this broad user community gives great incentives for hackers to attack the system. In 2019, Mika Veenstra, a scientist on the Defiant Threat Intelligence team, published that a malicious JavaScript has entered compromised websites. The aim was to create a new user with administrator rights on the victim's site. So, when an administrator logs in and views the infected site, he makes an AJAX call via jQuery, which then creates an administrator account with malicious privileges. According to Veenstra: “This AJAX call creates a user named wpservices with the email wpservices@yandex.com and the password w0rdpr3ss, with this user in place, the attacker is free to install further backdoors or perform other malicious activity.” Most likely (98%) and happening in the current case, attackers use WordPress plugins of third parties, which will be downloaded by the users to hack in. In 2019 around 55133 plugins were available and only 3 % of the plugins have been added in 2018. This leads to the assumption that a major part of the plugins are old, not updated and still in usage and ready to be hacked (Winder, 2019).

4.3 Security Concerns – Proprietary Software

It has been evaluated that open source software can be a hurdle or a blessing. For companies to decide whether open source or proprietary software fits better in their strategy, security concerns of proprietary software will be discussed further. One reason why many companies decide to use proprietary software is that the data of the company can be protected with legal and technical methods by technically detecting the source code from attackers. This makes it even harder for attackers to find out the code to write with the same functions. On the other hand, the company can also make use of legal methods, which often include intellectual property rights on the program.

Legally and technically, this gives companies the advantage to secure their data from outsiders, who at least initially will not have access (Swire, 2006).
One of the differences that can be evaluated is that overall proprietary software vendors are more organized than OSS communities. This leads to a higher integration of all processes, especially when it comes to the process of security checks in big software companies. Most commonly companies have modelled processes, which include threats and the implementation of security utilities. Thus, big companies like Microsoft conducts at least once in each product development cycle a safety audit. During this check, companies use code analysis tools to execute it on the source code and thus errors can be fixed. Overall, the whole source code is checked for security issues. In comparison to open source, the development processes are less formal and not always a security audit is conducted. As already mentioned above, everybody can see the source code, which means that not necessarily users with advanced knowledge see the source code and detect security errors. This might lead to the problem that even more errors stay undetected. Beyond that most likely the tools and skills big companies of proprietary software offer, are more progressive as the tools the OSS community have access to. Even though, the OSS community welcomes everybody to join, even without any information technical skills, big tech companies have invested huge amounts of money to boost the development of open source software. Companies like Red Hat and Intel etc. have employed full time developers to work on Linux (Clarke & Dorwin).

Another major point to consider is the fact that a company has (leaving out external testers) a monopoly, when publishing its proprietary software for the first time without overlaps of pre versions. This results in that only employees have all the information about the source code. If only the company has the information about the source code this also means that if vulnerabilities will be detected by the proprietary vendor itself, the vulnerability will be undisclosed and not available for the public. Thus, a vendor has to think about its strategy whether to disclosure the information about the vulnerability or not. If the proprietary software provider disclosure the vulnerabilities of the software, most commonly the vendor must expect that immediately sales will stop, and the reputation of the company will suffer from it. Overall, in the proprietary software industry this has led to the problem that most vendors have undisclosed their vulnerabilities. In order to prevent this concern, laws were written to disclosure

vulnerabilities to protect customers. This creates also the possibility for customers to directly address the vendor with the error. On the other side, disclosure can also have advantages as customers tend to trust the vendor as its disclosure’s vulnerabilities, which customers are dependent on. This leads to a higher trust and to a long-term quality reputation of the proprietary software vendor. Also, when disclosure errors of the software, external programmers can have access to it and make suggestions on how to solve the problem.
By making an impactful decision, the decision makers of the company seek for trust when making an investment. As above mentioned, proprietary software providers, who disclosure vulnerabilities will gain trust in the long-term relationship with their customers.
Considering this aspect that customers are more willing to buy from a proprietary software provider, who disclosure errors and is likely to accept help from external programmers the following assumption can be made. If a vulnerability is undisclosed and only known by an attacker, it will not be published, and the knowledge will only stay by the attacker. Thus, most commonly big companies, which buy software prefer the disclosure of such vulnerabilities. Otherwise, this would mean that worst case a customer can lost all his sales as only the attacker has knowledge about the vulnerability (Swire, 2006). Hence, it can be seen that a user of proprietary software is highly dependent on the proprietary provider. A company has to trust the vendor for reporting vulnerabilities in the software. This is especially from importance when confidential business information is involved.

4.4 Security Concerns – Volkswagen AG

After examining the downside of proprietary software, a well-known example will highlight the security concern of proprietary software in the industry on a real case. News about the Volkswagen exhaust gas scandal (VW) was spread all around the world. A part of their reputation in the US and worldwide fall apart because of the fraud and the aim of being the leader of manufacturing “Clean Diesel” cars around the world. Simply as it can be, VW used proprietary software to reach this goal but what they did was not legal in any way. VW has installed software in their cars that caused in the real-world higher emissions from diesel cars than in the tests they did before. In 2015 it has been determined that in the US, VW has used for their diesel cars (e.g. Käfer, Golf, Jetta and Passat as well as Audi A3) a software-operated device, which recognizes when emission tests will be conducted. During those tests the software restricts the emissions of the cars to fulfill the requirements of the US government. After the test was conducted the software and for normal usage of the cars, the cars emit up to 40 times the legal emission level. Thus, the Environmental Protection Agency (EPA) has VW assigned to fix these extra costs. But as this problem was not only solved with a software change, VW had to provide hardware modifications as well (Thurrott, 2015). In total approximately 11 million cars worldwide are affected (Focus, 2015). With open source software this fraud would have probably detected faster as with proprietary software.

4.5 Attacks in Proprietary and Open Source Software

In many cases it is difficult to differentiate between the question if open source is more secure than proprietary or the other way around. Thus, the following paragraph highlights two examples on which both kinds of software have to deal with security concerns. The first example is about the well-known “day zero” attack, which is a potentially serious software security weakness. Basically, it is a simple error in software but can cause great difficulties before the consequences are even discovered. A major problem is that zero-day attacks are not easy to detect. The attack starts as soon as the bug or vulnerability is exploited to introduce malware. This happens even before developers have implemented a patch to close or defense it. As developers are also humans, errors can occur, thus developers can create software that contains security errors without being aware about it. Most likely the attacker finds the vulnerability of the company and writes and implements exploit code while the vulnerability is still open. The best case for the company would be to find the attack as early as possible, as a business model has been established on the darknet to sell the hack. When an attack is published, developers start writing a patch to close the vulnerability, but to detect the error it can take years (FireEye). By considering the fact that most vulnerabilities stay over years undetected this gives a great opportunity for cyber operations, which do not have to be only done by attackers but also by governments or militaries (Ablon & Bogart, 2017).

Hence, stakeholders of companies may ask themselves what can actually be done to protect the company. The positive fact is that organizations can protect themselves in different ways. Firstly, by ensuring the software and operating systems are up to date. Secondly, individuals can choose Security Socket Layer (SSL) web pages with encryption of transmitted data. Lastly, companies should make use of including local area networks to secure the transmitted data (Kaspersky, n.d.).

Another typical attack of proprietary and open source is the brute force attack, which aims to hack a username or password or to discover a covert website as well as the key used to encode a message. This attack attempts of guessing the required information by the trial-and-error principle. Even though this kind of attack is outdated it is commonly used by hackers and is very efficient but can take, depending on the complexity of a password years to hack it. Thus, attackers have developed tools to fasten this process as for example, the most standard tool known as "dictionary attacks". Basically, the attacker is searching for a specific target and checks passwords with this specific username by using unabridged dictionaries and augment words with special characters and numerals.

Thus, it is from importance to have a powerful CPU because for example using a CPU, which can try 30 passwords per second would take more than two years to hack the password. Whereas adding a powerful GPU card would take around three to four days with 7100 passwords per second to hack the password with the same computer. This shows how easy it can be for an attacker to get your passwords.

By considering the fact that attackers with malicious intent have access to confidential data, it is necessary to know some tools to protect it. One possibility, which can help to overcome an attack is the functionality of login. Because if the user includes the function that the password can only be typed in three times before a lockout is activated for minutes, the attacker most likely switch to another user on which brute force attacks are easier to make. Another simple way to deal with it is that system administrators have to ensure their passwords are long enough - as more bits are included as more difficult it is to hack the password. These examples were explicitly important for IT specialists but as it is also important to protect the normal user, the following guidelines give a quick understanding on how to protect a "normal" users password. One simple and efficient way is to use a complex password and change this regularly. This password should include around ten characters, including symbols and numbers

(Kaspersky, n.d.). By considering both software types it can be seen that software in general is a main target of cyber-attacks. Hence, it does not always depend on which type of software someone is using and as most companies use both proprietary and open source software it is even more important to consider both.

5 Software Strategies

The chapter of different security concerns according to open source software, proprietary software and both has shown how vulnerable software can be when it comes to attacks. Thus, it is from even more importance to consider strategies, how companies can protect their data from unauthorized intruders. This chapter will give an overview what companies can do to be more secure.

As it is often difficult to differentiate between proprietary and open source software and many of the vulnerabilities are also shared, IT specialists can make use of the National Vulnerability Database (NVD), which is the most comprehensive and most collected database of published vulnerabilities. It was created by the NIST Computer Security Division, Information Technology Laboratory and was found in 2000 and since then the development of improvements take place constantly. According to the National Institute of Standards and Technology, the “NVD is the U.S. government repository of standard based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security related software flaws, misconfigurations, product names, and impact metrics.” Important to consider is, the NVD does not testing vulnerabilities, instead the database relies on third parties, mostly security specialists and vendors of software. After considering what it does not it is important to have in mind why the NVD is so successful and useful for companies. As the National Institute of Standards and Technology does not testing vulnerabilities, its employees execute and analyse reported Common Vulnerabilities and Exposures (CVE) by using the CVE Dictionary. This leads to the examination of vulnerability types and relevant statements etc., which can be used by companies to increase their software security (National Institute of Standards and Technology, n.d.).

For a company with vulnerabilities analyzed in the database the NVD can help a lot but on the other side, publishing vulnerabilities can also have negative side effects. Because attackers can easily make use of these published and analyzed vulnerabilities. Therefore, the NVD gives them the opportunity to screen vulnerabilities and seek information about which components are vulnerable and how the attack can be conducted. Openness is not always the most secure way.

5.1 Open Source Software Strategies

Specifically, in regard to open source, the following information will provide another example on how companies can secure their data. Due to the increasing use of OSS in the last years, the necessity of tracking components to protect companies from software vulnerabilities has become even more important. Normally, software development includes operating systems, leading to the problem that manual track is hard to conduct and requires scanning source codes, dependencies and binary data automatically. Thus, the Software Composition Analysis (SCA) is a solution that is particularly important to support the risk management, security and compliance with licensing requirements. SCA is the process of automating the visibility of the use of open source software. One major advantage of SCA is the possibility to receive a list of all components included in the applications, the license types and versions of components. This list is especially from importance for IT specialists to get a better understanding of the components used and leads to an increased knowledge about potential security vulnerabilities. As examined already, licensing is a relevant part regarding OSS and those companies using OSS have to comply to OSS licenses. Therefore, applying the Software Composition Analysis will help companies to become aware of certain necessities, respond to license compliance, establish policies etc. SCA enables proactive and continuous monitoring. For a better understanding, of workloads, SCA proactive monitors for security and vulnerability issues and enables users to generate significant alerts for recently discovered vulnerabilities in current and distributed products. These two examples have shown a small part of the advantages with conducting a SCA and as more than 50 percent open source code is compromised in software applications, the need for higher security standards is given. Thus, SCA can be a solution to generate higher security standards (Flexera, n.d.).

5.2 Proprietary Software Strategies

After examining a strategy on how companies can protect their data when using OSS, it is also necessary to focus on protecting proprietary software from hackers. Microsoft as one of the leading technology companies and producer of hardware and software in the world has the responsibility to guarantee a high standard of security. To ensure this, the Microsoft Security Response Center (MSRC), which is part of the security defense team, is developing for more than 20 years the security level of Microsoft. The following text excerpt states the mission of Microsoft:

"Our mission is to protect customers from being harmed by security vulnerabilities in Microsoft's products and services, and to rapidly repulse attacks against the Microsoft Cloud. We see this playing out in our everyday business by focusing on preventing harm, fast defense, and building trust in the community."

This statement gives a first impression how serious Microsoft sees security concerns for their business success and customer relationships. The problem with attackers is that it never stops, because even if a company has defended one attack successfully, the next attack occurs. Also, considering the fact that during the last years the amount of attacks has increased. This negative development of cyber-attacks can be seen in Figure 2 (Statista, 2020), which shows the increasing amount of monetary damage reported to the Federal Bureau of Investigation's (FBI) Crime Complaint Center (IC3). The aim of the IC3 is to report cyber-attack activity (Federal Bureau of Investigation , 2019). Overall this negative impact might highlight the fact that Microsoft has to continuously and dynamically engage in innovating new strategies for defending their products (Microsoft, n.d.). As the operating system Windows is programmed and developed by Microsoft, this product is a great example on how Windows increases its

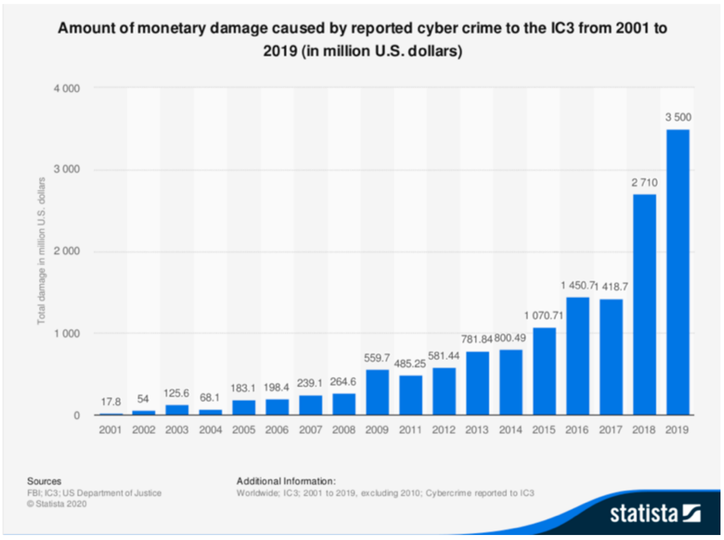

Figure 2: (statista, 2020)

security standard dynamically. Thus, for everybody who is familiar with Windows is most likely also familiar with the informally called activity “Patch Tuesday”. Every second tuesday in a month, Microsoft releases security updates for their proprietary software, which involve patches for vulnerabilities. And every fourth Tuesday of each month these updates come along with new features that are not only related to security. This routine helps especially network administrators to plan their schedules for testing’s and further software related activities. As the “Patch Tuesday” has become very successful, companies like Adobe copied the idea with also having a schedule (Bott, n.d.).

6 Microsoft and Open Source Software

Microsoft as the leading software and hardware company with over a million companies using Office 365 worldwide has changed its strategy from being the main proprietary software producer to becoming a leading edge in open source software (Liu, 2020). Today Microsoft is one of the main players when it comes to open source, but only a few years back in the past, Microsoft’s adjustment to open source was quite different. Because former CEO Steve Ballmer once called Linux and the GPL “cancer” (Tung, 2019). But after some time and further technological developments and the change in management this attitude has changed. In 2014, with the statement from the new Microsoft CEO Satya Nadella that “Microsoft loves Linux” the company has changed its strategy. In regard to the Azure Cloud of Microsoft, Nadella also admitted that already 20 percent of the operating systems on Azure are Linux and already contribute to Microsofts success story. Even though Microsoft Azure is still a proprietary software this has shown one example how Microsoft has changed its way towards open source. During the last years Microsoft contributed on several open source programs, for example, the open compute data center project of Facebook (Vaughan-Nichols, 2014).

6.1 Latest Contribution of Microsoft to Open Source Software

In 2019, Microsoft announced its collaboration with its competitor Linux to make Microsoft Teams also available for Linux users. The idea of Microsoft Teams is to bring users together via chat, video or calls. For Linux users, Microsoft Teams is now the first Microsoft 365 app, which can be used also via Linux. As it is in the interest of Microsoft to innovate their offerings, the collaboration with Linux is also a great chance for developers to increase their number of users. This big step from Microsoft shows exactly their intention against the original statement, comparing Linux with cancer towards a great collaboration with Linux. But not only Microsoft profits from this collaboration, also Linux does. To give an example on how Linux is positively convinced, Jim, Zemlin, Executive Director at The Linux Foundation has stated: “2019 has been another incredible year in open source, and Linux continues to be at the heart of all the growth and innovation. I’m really excited about the availability of Microsoft Teams for Linux. With this announcement, Microsoft is bringing its hub for teamwork to Linux. I’m thrilled to see Microsoft’s recognition of how companies and educational institutions alike are using Linux to transform their work culture.” (Salazar, 2019) Overall, with the shift in the C-Suite with CEO Satya Nadella Microsoft has made a huge progress towards open source software. Thus, it is even more important to also guarantee high security standards.

6.2 Microsoft – Open Source Security

After examining how Microsoft secures their proprietary software, the next paragraph will give a short overview how Microsoft emphasizes open source security. First of all, Microsoft is a major player in this field, the question is now, how customer benefits and how Microsoft approaches security concerns for their customers.

The three following main benefits states the benefits, which customers have by using open source:

1. A major benefit is the community, because bugs are reported, features are contributed and costs as well as the benefits are shared.
2. With open source, software can be developed faster as it is possible to connect components, which already exists. Instead of implementing them in a new way.
3. The overall quality is higher, as the focus is more likely on specialized software components.

With the following steps, Microsoft has stated how companies can reduce their risks when using open source software:

1. Important to reduce the risk of attacks and vulnerabilities is to know the components a company is using. This can be done, by automation tools like OWASP, Snyk and WhiteSource Bolt. For the check of the components it is necessary to include enough metadata.
2. After all components have been identified, it is important to check them for vulnerabilities. This can be done by several different checks such as using a commercial security intelligence, perform a static analysis, comprehensive security reviews and by ensuring no public vulnerabilities are contained (CVE's). Most likely the companies, which are using open source tools, can also make use of their provided vulnerability alerts. It also can help by monitoring the NVD for new vulnerabilities.
3. Another step to reduce security risks in a company is to always update the components. This is also from importance even though no vulnerabilities have been detected as some of the vulnerabilities have not been published and the risk to lose confidential data is more expensive and challenging than actually keeping the components up to date.
4. One main process, which each company should have when using open source software is the process of risk management. Thus, by detecting a vulnerability the stakeholders can use the strategy for a security response (Microsoft, n.d.).

By addressing these four steps when using open source software companies can reduce their risk of losing data by hackers.

7 Conclusions

In regard to the question whether open source software or proprietary software is more secure it can be said that both software types have their advantages and disadvantages.
In general, open source software includes for companies the great advantage that it involves less costs as proprietary software, however it needs employees with knowledge to implement it. On the other side, proprietary software comes usually with services and updates, nevertheless it has the negative effect of becoming depended. According to security concerns of open source and proprietary software it can be seen that when it comes to open source software, a large community does not directly imply that vulnerabilities will be detected and reported as also less skilled people can join the community. However, the fact that some proprietary vendors may hide vulnerabilities of fearing to lose their image, shows again the factor of being depended. Especially in times of further expansion of digital transformation, being dependent from only one company can be very risky as each company has to react as flexible as possible. Also, as with an increase in digital transformation and considering the fact of the current pandemic, attacks are increasing. It is from importance to have a clear understanding on how to react and which process steps will be conducted in case of an attack. Because knowing vulnerabilities will reduce the risk of losing confidential information. Thus, if a company has highly skilled IT employees it should consider using open source software as the source code can be modified and thus a company can react much faster. But to answer the question whether open source or proprietary software is more secure it can be said that there is no right or wrong answer to the question. It highly depends on how skilled the IT specialists according to open source software of a company are and which proprietary vendors will be considered. As big tech companies offer already regularly patches as well as the disclosure of vulnerabilities on their platforms, the risk can be reduced. But if a company does not emphasize security and does not update the program, each possibility offered is unnecessary. Overall, it can be said that only when a company put much effort on securing their software, both software types can be secure.

8 Bibliography

Ablon, L., & Bogart, A. (2017). *Zero Days, Thousands of Nights.* Rand Corporation.

Avner, G. (2019). *Your Guide to Open Source Vs Proprietary Software Security*. Retrieved April 20, 2020, from:https://resources.whitesourcesoftware.com/blog-whitesource/your-guide-to-open-source-vs-proprietary-code-security

Bart, J. (2018). *Increased security through open source .* Nijmegen: Institute for Computing and Information Sciences .

Bitkom. (2019). *Open Source Monitor - Studienbericht 2019 .* Retrieved May 15, 2020, from:https://www.bitkom.org/sites/default/files/2020-02/20200218_studienbericht-open-source-monitor-2019_0.pdf

Bott, E. (n.d.). *Insider's guide to managing Microsoft Patch Tuesday.* Retrieved March 21, 2020, from: https://www.techrepublic.com/article/insiders-guide-to-managing-microsoft-patch-tuesday/

Bouras, C., Kokkinos, V., & Tseliou, G. (2012). *Methodology for Public Administrators for selecting between open source and proprietary software.* Retrieved April 14, 2020, from: http://dx.doi.org/10.1016/j.tele.2012.03.001

Borchers, D. (2020). Sicherheit mit Hintertür. ct Innovative 2020, Heise Medien GmbH & CO. KG.

Brodsky, Z. (2018, February). *Git Much? The Top 10 Companies Contributing to Open Source.* Retrieved May 15, 2020, from: https://resources.whitesourcesoftware.com/blog-whitesource/git-much-the-top-10-companies-contributing-to-open-source

Clarke, R., & Dorwin , D. (n.d.). *Is Open Source Software More Secure?.* Retrieved May 19, 2020, from: https://courses.cs.washington.edu/courses/csep590/05au/whitepaper_turnin/oss(10).pdf

Cohan, C. (2003). *Software Security for Open-Source Syst*ems. The IEEE Computer Society. DOI: 10.1109/MSECP.2003.1176994

DiBona, S., & Cooper. (2005). *Open Sources 2.0: The Continuing Evolution*. O'Reilly Media; 1 edition (31 Oct. 2005)

EULA. (n.d*.). EULA template.* Retrieved May 13, 2020, from: https://www.eulatemplate.com

Federal Bureau of Investigation. (2019). *2019 Internet Crime Report.* Federal Bureau of Investigation.

FireEye. (n.d.). *What is a Zero-Day Exploit?.* Retrieved April 16, 2020, from: https://www.fireeye.com/current-threats/what-is-a-zero-day-exploit.html

Flexera. (n.d.). *The typical, modern software application is comprised of more than 50 percent open source code*. Retrieved March 29, 2020, from: https://www.flexerasoftware.com/blog/what-is-software-composition-analysis/

Focus. (2015). *Manipulationssoftware in 11 Millionen Autos weltweit im Einsatz*. Retrieved April, 14, 2020, from: https://www.focus.de/finanzen/news/vw-konzern-raeumt-ein-manipulationssoftware-in-11-millionen-autos-weltweit-im-einsatz_id_4964048.html

free software (why it is not feee) http://www.gnu.org/philosophy/free-sw.html free software foundation

Freist, R. (2020). *Die Souveränität wiederverlangen*. *IT & Karriere*. Heise Medien.

Github. (n.d.). Build like the best teams on the planet. Retrieved May 11, 2020, from: https://github.com/team

Hansen, H., Mendling, J., & Neumann, G. (2015). *Wirtschaftsinformatik.* De Gruyter.

Honsel, G. (2020*). Einmal Utopia und zurück .* Technology Review.

Internet Archive. (n.d.). *Why `Free Software' Is Too Ambiguous*. Retrieved March 28, 2020 from: http://web.archive.org/web/19991013111143/http://opensource.org/free-notfree.html

Kaspersky. (n.d.). *What's a Brute Force Attack?.* Retrieved May, 11, 2020, from: https://www.kaspersky.com/resource-center/definitions/brute-force-attack

Kaspersky. (n.d.). *What is Zero Day Exploit.* Retrieved May 11, 2020 from: https://www.kaspersky.com/resource-center/definitions/zero-day-exploit

Landy, G., & Mastrobattista, A. (2008). *A Pragmatic Guide to 9 Open Source. In A. J. Gene K. Landy, The IT / Digital Legal Companion: A Comprehensive Business Guide to Software, IT, Internet, Media and IP Law*. Syngress; 1 edition.

Lawton, G. (2002). *Open Source Security: Opportunity or Oxymoron?. DOI: 10.1109/2.989921*

Liu, S. (2020). Number of Office 365 company users worldwide as of February 2020, by country. Retrieved May 15, 2020, from: https://www.statista.com/statistics/983321/worldwide-office-365-user-numbers-by-country/

McGovern, N. (2019, December). *GNOME*. Retrieved April 10, 2020, from: https://blogs.gnome.org/engagement/2019/12/31/why-we-need-a-free-desktop/

Mehta, D. M. (n.d.). *Effective Software Security Management.* Retrieved April 27, 2020, from: https://owasp.org/www-pdf-archive/Effective_Software_Security_Management.pdf

Microsoft. (n.d.). *Microsoft Seucirty Response Center*. Retrieved May 14, 2020, from: https://www.microsoft.com/en-us/msrc/mission?rtc=1

Microsoft. (n.d.). *Open Source Security*. Retrieved May 14, 2020, from: https://www.microsoft.com/en-us/securityengineering/opensource?activetab=security+analysis%3aprimaryr3

Mouelhi, T., El Kateb, D., & Le Traon, Y. (2015). *Inroads in testing access control.* from: Advances in Computers, Volume 99, DOI: 10.1016/bs.adcom.2015.04.003

National Institute of Standards and Technology. (n.d.).*National Vulnerability Database.* Retrieved April 25, 2020, from: https://nvd.nist.gov/general

Online Etymology dictionary. (n.d.). Retrieved March 18, 2020, from. https://www.etymonline.com/word/proprietary

Open Source Initiative. (n.d.). *Open Source Initiative*. Retrieved March 21, 2020, from: https://opensource.org

Optimus Information. (2015). *Open-Source vs. Proprietary Software Pros and Cons*. Retrieved April 16, 2020, from: http://www.optimusinfo.com/downloads/white-paper/open-source-vs-proprietary-software-pros-and-cons.pdf

Payne, C. (2002). *On the security of open source software .* Murdoch: School of Information Technology.

Sahoo, R., & Sahoo, G. (2016). *Computer Science with C++.* New Saraswati House (India) Pvt. Ltd.

Salazar, M. (2019). *Microsoft Teams is now available on Linux*. Retrieved May 15, 2020, from: https://techcommunity.microsoft.com/t5/microsoft-teams-blog/microsoft-teams-is-now-available-on-linux/ba-p/1056267#

Sametinger, J. (2013). *Software Security.* Linz : Johannes Kepler University. DOI: 10.1109/ECBS.2013.24

Schneider, F. (2000). *Open Source in Security: Visiting the Bizarre.* DOI: 10.1109/SECPRI.2000.848477

Stallman, R. (2002). *Selected Essays of Richard M. Stallman, 3rd Edition.* Free Software Free Society.

Statista. (2020). *Amount of monetary damage caused by reported cyber crime to the IC3 from 2001 to 2019*. Retrieved May 19, 2020 from: https://www.statista.com/statistics/267132/total-damage-caused-by-by-cyber-*crime-in-the-us/*

Swire, P. (2006)*. A Theory of Disclosure for Security and Competitive Reasons: Open Source, Proprietary Software, and Government Systems. Houston Law Review, Vol. 42, Issue 5, 2006 January 31*, 2006

Thurrott, P. (2015). *Volkswagen Used Software to Cheat on Emissions.* Retrieved April 28, 2020, from: https://www.petri.com/volkswagen-used-software-to-cheat-on-emissions

Tung, L. (2019*). Richard Stallman to Microsoft: Publicly retract 'open source is a cancer' claim*. Retrieved April 5, 2020, from: https://www.zdnet.com/article/richard-stallman-to-microsoft-publicly-retract-open-source-is-a-cancer-claim/

Vaughan-Nichols, S. J. (2014). Why Microsoft loves Linux. Retrieved May 9, 2020, from: https://www.zdnet.com/article/why-microsoft-loves-linux/

White Source. (2020). *The State of Open Source Security Vulnerabilities - White Source Annual Report.* Retrieved May 2, 2020 from: https://www.whitesourcesoftware.com/open-source-vulnerability-management-report/

WhiteSource. (2020, February). *The Complete Guide for Open Source License.* White Source. Retrieved April 6, 2020, from: https://resources.whitesourcesoftware.com/licenses/the-complete-guide-for-open-source-licenses-2020

Wiedemann, A., Holey, T., & Wiedemann, A. (2007). *Wirtschaftsinformatik.* Klaus Olfert. Kiehl.

Winder, D. (2019, August). Critical 'Backdoor Attack' Warning Issued For 60 Million WordPress Users. Retrieved May 1, 2020, from: https://www.forbes.com/sites/daveywinder/2019/08/31/critical-backdoor-attack-warning-issued-for-60-million-wordpress-users/

WordPress. (2020). *A live look at activity across WordPress*. Retrieved May 4, 2020, from: https://wordpress.com/activity/

Zhang, Y., Malhotra, B., & Chen, C. (2018). *Industry - Wide Analysis of Open Source* Security. DOI: 10.1109/PST.2018.8514185