UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA
(FINAL PROJECT PAPER)

KIBERNETSKA VARNOST IN RANLJIVOST PROGRAMSKE OPREME
(CYBERSECURITY AND COMMON SOFTWARE VULNERABILITIES)

DAMJAN DIMITROV

2021

ZAKLJUČNA NALOGA (FINAL PROJECT PAPER)

DIMITROV

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

(Final project paper)

Kibernetska varnost in ranljivost programske opreme

(Cybersecurity and common software vulnerabilities)

Ime in priimek: Damjan Dimitrov
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Peter Rogelj

Koper, avgust 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Damjan DIMITROV

Naslov zaključne naloge: Kibernetska varnost in ranljivost programske opreme

Kraj: Koper Leto: 2021

Število listov: 71 Število slik: 12 Število tabel: 1

Število referenc: 35

Mentor: doc. dr. Peter Rogelj

Ključne besede: kibernetska varnost, informacijska varnost, programska oprema, spletna ranljivost, spletni napadi, Internet

Izvleček:
V tej diplomski nalogi predstavljamo področje kibernetske varnosti vključno z aktualnimi cilji, standardi, načeli, dobrimi praksami in priporočili. Kibernetske napade klasificiramo v več skupin in predstavljamo najpogostejše vrste kibernetskih napadov z opisom njihovih najpogostejših oblik in njihovimi posledicami. V nalogi poudarimo tudi pomen neprofitnih organizacij, ki delujejo na področju kibernetske varnosti, in predstavljamo fundacijo OWASP z njenimi cilji in trenutnimi projekti. Izpostavimo projekt "OWASP top 10", ki je stalno posodabljana zbirka desetih najbolj kritičnih spletnih groženj.
V drugem delu naloge se posvetimo praktičnemu preverjanju spletne varnosti specifične javno dostopne spletne storitve. Identificiramo in opišemo njena varnostna tveganja in kategoriziramo s tem povezane grožnje

Key document information

Name and SURNAME: Damjan DIMITROV

Title of the final project paper: Cybersecurity and common software vulnerabilities

Place: Koper Year: 2021

Number of pages: 71 Number of figures: 12 Number of tables: 1

Number of references: 35

Mentor: Assist. Prof. Peter Rogelj, PhD

Keywords: cybersecurity, information security, software, vulnerabilities, cybercrime, cyber crime, cyber attacks, Internet

Abstract:
In this thesis we present the field of cybersecurity, including its current goals, standards, principles, good practices and recommendations. We classify cyber attacks into several categories and present the most common types of cyber attacks with a description of their most frequent occurrences and consequences. The thesis also emphasizes the importance of non-profit organizations working in the field of cyber security, and we present the OWASP Foundation with its goals and ongoing projects. Highlighted is the “OWASP Top 10” project, which is a constantly updated collection of the ten most critical online threats.
In the second part of the thesis, we focus on the practical verification of the online security of a specific publicly available online service. We identify and describe its security risks and categorize related threats.

ACKNOWLEDGEMENT

I would like to capture this opportunity to express my gratitude towards the University of Primorska, principally to the Faculty of mathematics, natural sciences and information technology (FAMNIT) for lending me the privilege to be a foreign student in Slovenia and pave my path towards further success and new accomplishments.

I am thankful to all my professors who have always been examples of knowledgeable mentors anyone would look up to, most specifically my thesis mentor doc. dr. Peter Rogelj, who has been of the most help during working on my thesis and was always available to provide me with guidance.

Lastly, I would like to say a big thank you to my family for always being endlessly supportive to me, especially my parents and my girlfriend who have never stopped being a continuous inspiration and motivation to me.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
TLS	Transport Layer Security
URL	Uniform Resource Locator
HTML	HyperText Markup Language
DNS	Domain Name System
SQL	Structured Query Language
2FA	Two-Factor Authentication
JSON	JavaScript Object Notation
CSRF	Cross-Site Request Forgery
DTD	Document Type Definition
CI/CD	Continuous Integration / Continuous Deployment (Delivery)
LDAP	Lightweight Directory Access Protocol
SMTP	Simple Mail Transfer Protocol
ORM	Object Relational Mapping
OS	Operating System
XML	Extensible Markup Language

1 INTRODUCTION

In the last decade information systems and networks have become an integral part of our daily lives. Apart from individuals, many organizations - financial, medical, educational, corporate and more, operate with networks and information systems on a daily basis in order to function more effectively. Due to the overwhelming occurrence of data gathering, sharing, storage and networking, concurrently came afloat the necessity for the stored and transferred data to be secured, protected and it has been made transparent how much security in software is important. Making sure a software is secure, safe and invulnerable to breaches has proven to be sufficiently necessary to become a field of study on its own. Software security has become a subfield or a branch of computer science, aiming to improve security and safety in any type of software.

Techniques of security and safety which apply to real life are very similar to the ones which apply to software. The same principle of things which remain protected, private and inaccessible to anywhom have the same value and principle when presented in the digital world. As attackers rely on finding existing or new flaws in a computer software, the field of computer security is focusing on finding a way to solve vulnerabilities and explore methods to prevent them from happening.

Improperly secured software can lead to data breach, data loss, unauthorized access, elevation of privilege, stolen information, damage to the codebase or may even harm the system and network. For this reason, this thesis focuses on studying and exploring the most common contemporary types of software vulnerabilities, how we can prevent them and stay protected from them.

In the following section we will get more acquainted with the aforementioned field of study related to software security - also known as ***Cybersecurity***.

2 CYBERSECURITY

Cybersecurity, also known as Computer Security, Software Security, or Information Technology Security is a collection of techniques which involve the protection and securing of various digital components such as data, applications, networks and computer systems from unauthorized access, theft or damage. In the 1950s, the word “cyber” used to refer to cybernetics – the science of understanding the control and movement of machines and animals. This was followed by “cyber” standing for “computerized”. The 1990s brought around a new cyber-related term - the word “cyberspace” emerged to define an invented physical space that some people wanted to believe existed behind the electronic activities of computing devices.
Not only is cybersecurity a major contemporary and rapidly growing field of study, it is also a large ongoing effort to protect systems and networks from potential harm or breach. On a substantial level, the aim of cybersecurity is to distinguish and determine harmful pieces of software, find a resolution for the appearing issues and make sure they will be avoided in the future. A component of some software which can be exploited is also known as a vulnerability.

A software vulnerability represents a flaw or a weakness in the design, implementation or functionality of a certain component belonging to that software. A vulnerability can often be exploited and used as an entry point for the attackers to be able to abuse the software, such as stealing information or gaining privileged access to other components. Vulnerabilities can occur on many occasions - in the software itself, computer networks, third-party libraries or packages imported in the software, all of which can be either researched, reverse-engineered, probed or exploited by using several attacking techniques, scripts and tools [1].

From an elevated point of view, cybersecurity does not concern itself solely with vulnerable parts of a software. It also exists in order to establish numerous rules, standards, principles, practises and policies which need to be followed when designing and developing information systems. A good example for this is HTTPS, which was added as an extension to HTTP in order to set up a secure and encrypted data transfer over a network. Following is HTTP STS (Strict Transport Security) which restricts a website to only allow communication with external resources only if they use the HTTPS via TLS protocol. Such and many more regulations are one of the primary concerns of cybersecurity.

2.1 Why we need cybersecurity

Properly securing information and data is crucial in many ways. The information which is required to be protected may vary depending on what the information holds and represents, how many people require the data, how important the information is and how private it should be kept.

On a personal level, the data which needs to be safeguarded is a person's identity and his personal information such as full name, age and ID number. When a person is registered as a user of some service, there can be potentially a lot of confidential information which must not be disclosed or exposed - this can vary from a person's email and phone number to the person's credit card number or medical records, depending on the type of service the user is using. On an organizational and corporate level, confidential information includes financial records, employee information, corporate agreements and more. Breach or theft of this kind of information may lead to a very large damage to the company's network or software, its employees and customers and stain the company's reputation. Lastly, guarded information may be on a level as big as an entire state or country, where improper securing of such information may present a harm to national security and the well-being of the country and its citizens [8].

The digital space has also provided a new dimension to warfare, where nations can carry out conflicts without physically leading a war the traditional way. This allows countries with minimal military presence to be as strong as other nations in cyberspace. Cyberwarfare is an Internet-based conflict that involves the penetration of computer systems and networks of other nations. These attackers have the resources and expertise to launch massive Internet-based attacks against other nations to cause damage or disrupt services, such as shutting down a power grid [2].

In order to prevent such horrible scenarios from happening, the field of cybersecurity increasingly and consistently researches methods and regulations to impede cyber attacks.

2.2 Common types of cyber attacks and infiltration techniques

A cyber attack is an offense that is directed against a person, organization or a larger institution by means of cyberspace. Cyberspace, a virtual space that does not exist,

has become the metaphor to help us understand digital weaponry that intends to harm the people. Although the people who own the digital devices or platforms are victims, they are usually only the implicit targets of cyberattacks, while the attack is explicitly executed towards the devices, software, systems or computer networks. Despite the harm done through a cyberattack is not physically tangible, what is real however, is the intent of the attacker as well as the potential impact. While many cyber attacks are merely nuisances, some are quite serious, even possibly threatening entire businesses, organizations or even states.

As time goes by and technology keeps advancing, growing and reaching new heights and milestones, similarly the need to put a larger emphasis for security on any existing and new technologies expands. Unfortunately, as cyberspace is widening and cybersecurity is becoming more sophisticated, cyber attackers, also known as hackers, keep discovering new types of security exploits at a similar rate. The more complex a software or technology is, the bigger the chance for a vulnerability to take place, which prompts the need for consistent analysis, probing, verification and re-establishment of security rules, practises and implementations. It is already common knowledge that in order to fix a software vulnerability or prevent hackers from abusing it, the engineers need to be well aware of potential and possible issues and means of breach beforehand.

In the following subsections we will go through a couple of the most common types of cyber attacks, based on quantitative analysis from the past 10 years.

2.2.1 Malware

Malware is one of the most commonly occurring types of cyber attacks. It is short for Malicious Software and it represents a script or a program which is executed on the victim’s system or device, after which it can perform many kinds of malicious actions, depending on the type of the malware. It can be used to cause harm, steal data, bypass or even compromise an entire system. It is most often delivered on a target system via a malicious email, clicking on an unknown link or URL, or downloading a file from an unofficial website. Malware has actually been a threat to individuals and organizations since the early 1970s. Since then, the world has been under attack from hundreds of thousands of different malware variants, all with the intent of causing the most disruption and damage as possible [3].

Due to the fact that malware is very wide-spread, over the years many new classifications and variants of malware have appeared, and for ease of distinguishment they have been separated into different classes of malware:

2.2.1.1 Ransomware

Ransomware works in such a way that it takes on complete control of the target's operating system - usually completely locking the end-user out of their own system without the user being able to access it. It also encrypts all of the data and files stored on the system. Following this scenario, the attackers demand a ransom from the target if they wish to have their data and access restored. Usually targets of ransomware are predetermined targets by the hackers, who analyze the victim and intend to have financial gain from the ransomware attack [3].

2.2.1.2 Spyware

Spyware is a type of malware which usually does not do explicit harm to an end-target's system or device, yet it is used to track all kinds of activity a person is performing on the system, such as monitoring keystrokes, visited websites, saved files, etc. It is most often used to retrieve sensitive and/or confidential information about a person, such as personal secrets, private data and information such as credit card number, personal ID number and more. It often modifies or overwrites the system's firewall, antivirus and general system security settings in order to not be identified on the system [8].

2.2.1.3 Adware

Adware has the purpose of infecting an operating system or a program inside the operating system in order to persistently present advertisements while running. This type of malware does not exactly do significant harm to the system, however it is very bothersome to the user who is using the infected system. It is often bundled together with other kinds of more harmful malware, usually to perform a distraction because of its self-evident and disruptive nature.

Adware can also present another type of malware or virus which has infected a system through an user clicking on an advertisement and therefore visiting a malicious website or downloading the malware through opening the advertisement.

For this reason, advertisements which offer rewards, very high discounts, or sales which are not related to the current website or service should be ignored and preferably removed through an ad-blocker [5].

Figure 1: Adware visualization
(Source: https://vpnoverview.com/internet-safety/malware/adware/)

2.2.1.4 Scareware

This is a type of malware which has an intention to persuade the user into performing a specific action such as opening a certain link or downloading a file based on the user's fear. The malicious software manifests as flashing pop-ups or dialogue windows, informing the user of a very serious effect which will take place if a user does not immediately perform an action, or a similar prompt which represents fabricated information in order to stem fear inside the end-user's mind. Scareware is in fact used so that a user can perform an action which opens an entrance to another potentially more harmful malware [5].

2.2.1.5 Virus

A virus is possibly the most common malware of all and they are usually infecting a system via an executable file. The name is analogous to a biological virus due to its potential to spread and cause further harm. A virus on a computer can only have a disruptive purpose, such as slowing down the whole system or disturbing the user,

although they may also impair the system by randomly deleting or moving files and data. Usually the sluggish operation of a device, the constant freezing and crashing of programs and the visible burden on the device's hardware are very quickly correlated to a virus being present. The virus is capable of replicating and acting on its own will. Some more advanced viruses are able to inject code into the operating system and rewrite certain rules and restrictions. Computer viruses can spread through several infection mechanisms over the internet, however mostly it is through other types of malware or social engineering. This includes malicious emails, download files from unofficial websites, advertising, or also using an outdated (deprecated, obsolete) version of an operating system which is no longer under support by the maintainers [9].

2.2.1.6 Trojan Horse

The trojan horse malware respectfully holds that title for the reason that it carries out malicious operations under the disguise of a regular and authorized program. This type of malware is one of the most dangerous and crippling, because it can exploit the privileges of a user that is currently using the operating system, as well as lifting privileges from the user and creating backdoors for attackers to enter the system. Another reason this type of malware is so dangerous is that it can be contained not only in executable files or folders, but also basic files such as text files, audio files or images. Similarly to malware, the trojan horse has multiple subcategories based on its intent and infiltration type. Antiviruses help a lot when it comes to detecting and removing trojan horse malware from a system. Nevertheless, the same safety rules and principles should be followed to avoid being infected by this malware [10].

2.2.1.7 Worm

Worms also belong in the group of some of the most destructive types of malware there are. They are not bound to a host program and can run deliberately. The worm malware is endlessly self-replicating and has the purpose of slowing down the target's system or even the entire network of systems. Due to its quick replicating and indeterministic nature, the worm is very hard to be completely removed from a system and usually leaves the target with the need to completely wipe the system from its contents. Computer worms make use of some of the deepest and most

dangerous vulnerabilities in a victim's computer. In order to propagate itself further, it then follows known holes in networking and file transfer protocols.

A worm may not do any damage at all: in the early days of computing, worms were sometimes designed as larks or proofs of concept to exploit security holes, and did nothing more to infected computers than reproduce themselves in the background. Often the only way to know anything has gone amiss is when the worm has made too many copies of itself on a single system and slowed down its operations. Though as operative systems' security improves and writing a worm that could crack it gets more difficult and takes more resources, worms have become a means to an end. Today, worms almost inevitably include payloads - code that carries out some larger malicious purpose beyond the reproduction and propagation of the worm itself [11].

2.2.1.8 Rootkit

This malware is designed specifically to modify and overwrite an operating system's security rules and precautions, disable antiviruses and create a backdoor which the attacker can access to gain control of the target's system. The rootkit works with the foundation of gained and elevated privileges of the device, while simultaneously lifting the privileges and permissions from the user who is using the system. It is also common for rootkits to modify system forensics and monitoring tools, making them very hard to detect. Often, a computer infected by a rootkit must be wiped and reinstalled [8].

2.2.2 Phishing

Phishing, a homonym to the word fishing and also analogous to it, is a type of cyber attack which takes place when a hacker is trying to steal data or prompt a user into opening a fake link, website, downloading a file, or sometimes even paying real money. Phishing is very closely related to social engineering. By definition, Social engineering is the act of tricking or persuading someone into exposing information or taking action, usually through technology [6].

There are several factors in social engineering which attackers abuse in order to force the victim to do what the attacker desires. Most often, the attackers are abusing a victim's ignorance and emotional response to critical and urgent information. For example, an attacker might send an email to some internet user in

which the victim is informed that he has been a victim of a major security breach, whereas he needs to immediately fill a form in order to be safe from the attack. In panic, the user does not take the time to consider whether the email is ill-advised or not and enters the required information, which is then exposed to the hacker. Another type of phishing attack may come internally within a company or an organization, where via a fake email very closely (though not entirely) resembling the company's domain, sends an e-mail to the whole company informing them of an urgent problem, claiming that in order to resolve the problem they urgently need to download a file attached to the email. Unknowingly and unsuspiciously, the people who have received the email had downloaded the file and by doing so they have welcomed a malicious virus into their system, which had infected and harmed the entire company's network.

Cyber criminals most often perform phishing in very similar scenarios, by abusing the victim's ignorance and inducing a sense of urgency and persuasion into their message. In order to be wary of such attacks, users need to prepare a couple of security and fact checks before acting on the requested action. The most important rule at first glance is to verify the validity of the email sender's address, whether it comes from the appropriate email domain or not and always review the underlying address of hyperlinks before opening them. Another thing is to ask oneself the question of whether the information presented is truly plausible or it is actually "too good to be true".

Due to this type of cyber attack being very widespread, common and present on a large level, many non-government organizations exist who have the purpose of spreading the knowledge about this type of attack and teaching people how to be aware and prevent the occasion where they fall victim to phishing [12].

2.2.3 Man In The Middle Attack

The man in the middle attack is sometimes also classified as a type of malware. The attack has earned that name due to the reason that the hacker acts as an interceptor for the traffic relay which is sent and received from a target device. This way, the attacker has complete control and monitoring of the data which travels to and from the victim's device without the victim's knowledge or awareness, while in the meantime being able to alter it willingly.

Apart from monitoring and relaying the communication channel and being capable of stealing any information, the attacker is at the same time capable of

sending fabricated information, tricking the unknowing user into thinking that he/she is communicating through the internet via a private connection, when in reality the attacker is controlling the whole plot. This type of attack is very closely related to eavesdropping and belongs to the sensitive data exposure category of cyber attacks. It most often happens when users connect to insecure publicly available Wi-Fi hotspots or web pages who haven't secured their internet traffic. The attackers can mimic these connections with the same name or domain and without requiring a password to connect to them, which marks a user as a potential victim when connecting to that network.

In order to stay safe from a man in the middle attack, the HTTP protocol has been replaced with a newer version with updated security mechanisms, also known as HTTPS (S for secure). However, there are still some scenarios where HTTPS does not help entirely, such as:

- Requesting a website, where the first request per default is an HTTP request
- Requesting an HTTP page from the server
- Server redirects to HTTPS latterly
- Man in the middle requests HTTPS page
- The server responds with an HTTPS response
- The interceptor rewrites HTTPS to HTTP and can inject malicious code
- The clients future requests are now HTTP can now be sniffed and tampered with by the man in the middle [13]

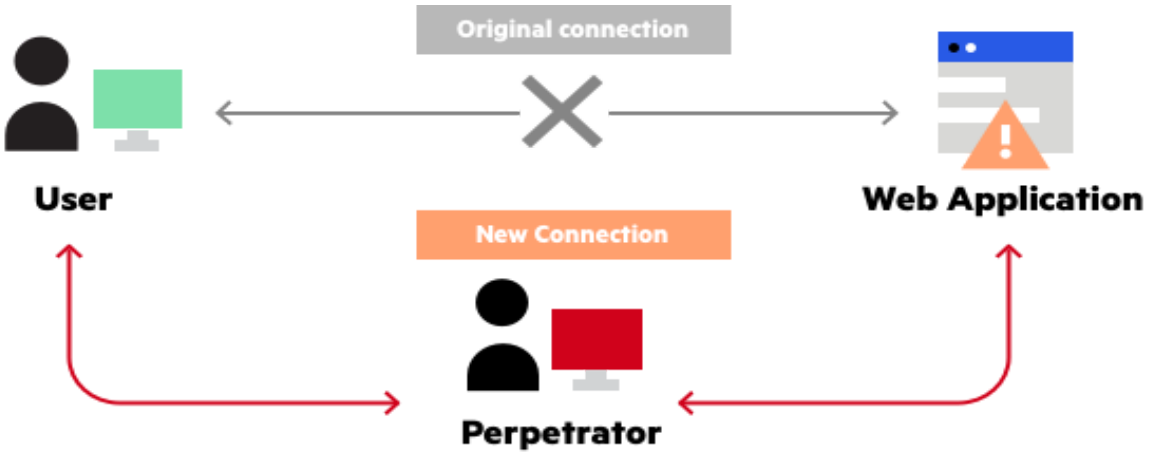

Figure 2: Man In The Middle attack visualization
(Source: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/)

To prevent this attack from happening even in these scenarios, there exists a mechanism called HTTP Strict Transport Security (abbreviated as HTTP STS, or HSTS), which sets the HTTP Strict-Transport-Security response header and lets a website inform browsers that it should only be accessed using HTTPS, instead of using HTTP. This header does not allow a website to be loaded through HTTP and then redirected to HTTPS, because this gives hackers space to perform the attack by intercepting the initial HTTP request. If the HSTS header is present, the website does not allow it to be loaded through HTTP at all.

Another solution to go around the man in the middle attack is a VPN (virtual private network), which provides privacy, anonymity and security to users by creating a private network connection across a public network connection. It proxies a user's connected network to a different network than the one to which the user is currently connected to.

2.2.4 Poisoning

Poisoning in cybersecurity refers to a type of attack related to fabricating a record, claim, information or tool such that a hacker can gain leverage over his victims by mimicking a fraudulent entry point as genuine. The attack relies on the attacker being able to intercept or rewrite some stored data, or can also involve social engineering techniques to deceive a user or an algorithm. The most infamous types of poisoning are SEO poisoning and DNS cache poisoning.

2.2.4.1 Search Engine Optimization (SEO) Poisoning

Search engine optimization is observed as a technique in which an attacker wants to decept the search engine's algorithm into showing the attacker's page as trending or popular, meaning it is displayed among the first in the list of results from the search query. The intention behind this approach is to draw as much attention as possible into the attacker's own website which can be identical to another website or a unique website with potential malware hidden under its hood. Another similar scenario is where the attacker does the same, however not for their own website and domain, instead to another user for which the attacker has already exploited a vulnerability which might introduce malware into many users' systems.

In the earlier stages of search engines, attackers were able to abuse the search engine's algorithm due to the reason that search engines still required further development and improvement to overcome such issues. The point was to force the malicious website to point to as many external pages as possible, as well as the opposite. Attackers were easily able to achieve this by adding hidden referenced anchor tags to the website. Nowadays, attackers achieve similar results by doing the same as well as adding keywords, external links and alternatives and ingenuine accessibility elements to the HTML tags of the website in order to drive it above in the search query result list [14].

2.2.4.2 DNS Cache Poisoning

DNS stands for domain name system and it is a system whose purpose is to associate IP addresses with domain names. The DNS is the reason why we have human-readable domain names such as google.com, instead of having to issue a request to 172.217.168.174 each time we wish to go to Google's home page. When we search for google.com, the DNS record is verified and the domain name is mapped to its respective IP address, after which the server provides the user back with the response from the domain's IP address.

DNS Cache Poisoning, also referred to as DNS Spoofing, is a flaw into the domain name system when an attacker can inject a malicious IP address into it. It works in such a way that the attacker tampers with the DNS Cache, where the records of already visited websites are stored along with their time to live (TTL). The TTL informs the DNS about how much more time the record will remain cached in the storage. Until the TTL has still not expired, the websites a user has recently browsed are stored in the cache so that there is no need to query multiple higher-level domains each time a user queries for a specific website. Inside the DNS cache, an attacker falsifies the information of the pointing IP address of a web address, such that when a user re-opens the website he is directed towards the attacker's URL which may contain malware.

Similarly to SEO poisoning, the attacker might recreate a duplicate of a website, with the difference that the hacker's website contains malicious code, however the user is not going to be aware of this at first glance and is a potential victim of the DNS poisoning attack [15].

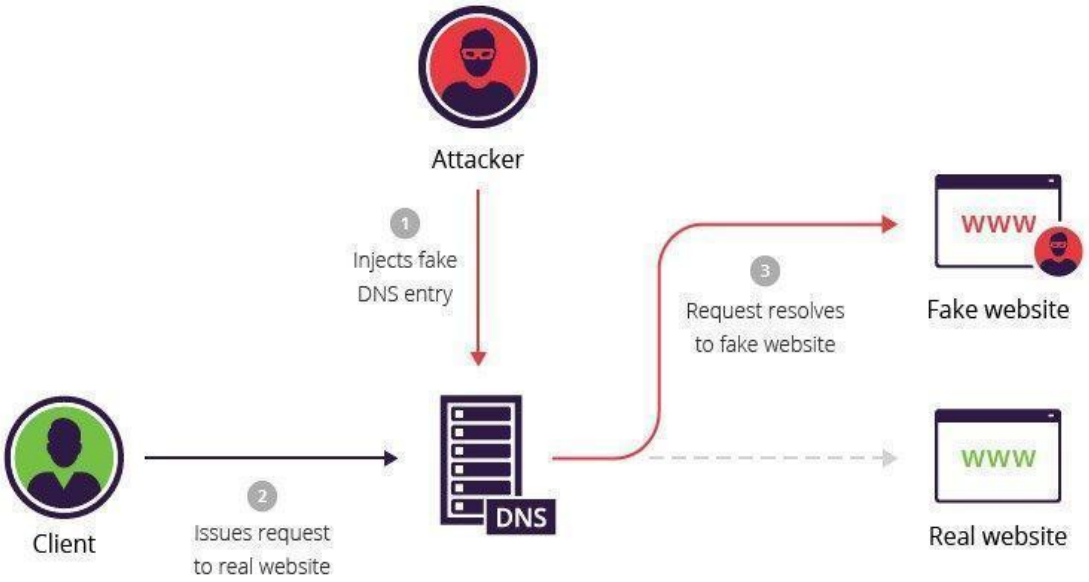

Figure 3: DNS Poisoning visualization

(Source: https://www.imperva.com/learn/application-security/dns-spoofing/)

2.2.5 Denial Of Service (DoS)

A denial of service attack is a type of network attack which causes disruption to a network, service, or server. While a system is under a denial of service attack, it is temporarily unresponsive and access to it is partially or completely shut down due to overloading the network or the server with requests or bogus data.

Denial of Service attacks are considered a major risk because they can easily interrupt communication and cause significant loss of time and money. These attacks are relatively simple to conduct, even by unskilled attackers. A measure to stay safe from a DoS attack is possibly to implement an automatic network/server timeout and reboot, after it has been inactive for a while or it has been constantly receiving requests for a certain amount of time. This refreshes the frozen state of the network while it is under attack. It is crucial to have established a strong network infrastructure and a response team/plan for a potential appearance of this attack. It is also always a good idea to retain server logs of the previous network requests in order to perform worthwhile forensic analysis [2], [4].

The DoS attack is commonly categorized in three varieties:

2.2.5.1 Overwhelming Quantity of Traffic

This is when a network, a server, or a host is being sent an enormous quantity of data or network requests at a rate it cannot handle, over a very small time frame. This can cause a major slowdown or in severe cases a complete disruption and crash of the target system. After this attack it might take a whole day for the attacked system to recover. For this type of denial of service a key point of how the network handles the attack is the integrity of its infrastructure.

2.2.5.2 Maliciously Formatted Packets

In this classification of the DoS attack an intentionally damaged network packet is sent to the host, rendering the receiving server unable to process and handle the packet. This results in the system to start running very slowly because of redirecting the processing load into treating the packet, or may sometimes also result in a system crash.

2.2.5.3 Distributed Denial Of Service (DDoS)

This is the most nefarious, infamous, dangerous and harmful attack of all three categories of denial of service attacks. A distributed DoS attack is quite similar to DoS, however it originates from multiple coordinated sources. It works in such a way that the attacker first has a preparation phase to launch this attack, during which he is randomly finding victims across the world whose devices he infects. These hosts are usually called bots or zombies and the network of infected hosts is called a botnet.

The people the infected devices belong to are not aware that they contain the malicious script inside their system, but once ready, the attacker launches the coordinated distributed attack from all the infected devices, which substantially increases the amount and heaviness of the data payload which is being spammed to the targeted network. This attack may also be issued to multiple target networks simultaneously, due to its integrity formed by the large number of requesting clients.

Sometimes the consequences of such an attack can be quite severe and it takes a long time for the targeted system to recover to its original state.

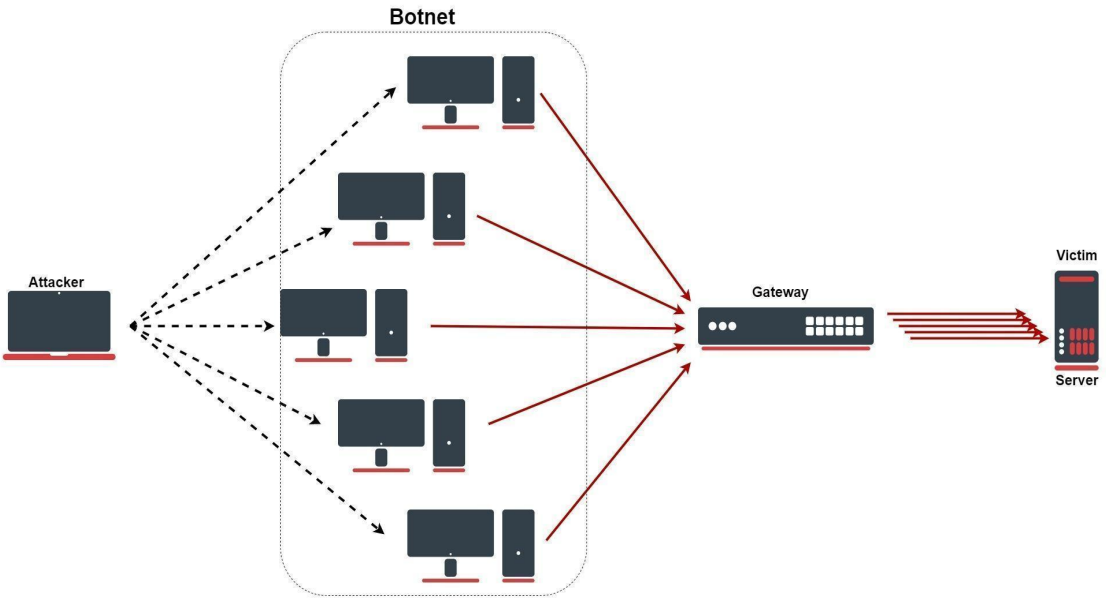

Figure 4: Distributed Denial of Service attack visualization
(Source: https://help.mikrotik.com/docs/pages/viewpage.action?pageId=28606504)

2.2.6 SQL Injection

SQL Injection is a code injection technique that attackers use to attack data-driven applications. The attack is most often executed in an input fashion, allowing the hacker to inject malicious SQL queries as input and send that input as a query parameter or a header to the server. This attack falls in the category of Injection attacks, which we will look at in more detail in section 3.1.1.

The problem that the attackers exploit when executing the SQL injection attack lies in the fact that the input the application receives is not sanitized and is directly interpolated into a hardcoded SQL query which is usually sent to the server in order to retrieve data from the database. By abusing this problem, the attacker is able to deliberately retrieve any data from the database, as much as the whole table of records which is being requested from the server. More skilled hackers are able to take this further and ruin the database, such as deleting records, changing database permissions, changing database transaction balances, or even exfiltrating the entire database schema along with the data inside it [7].

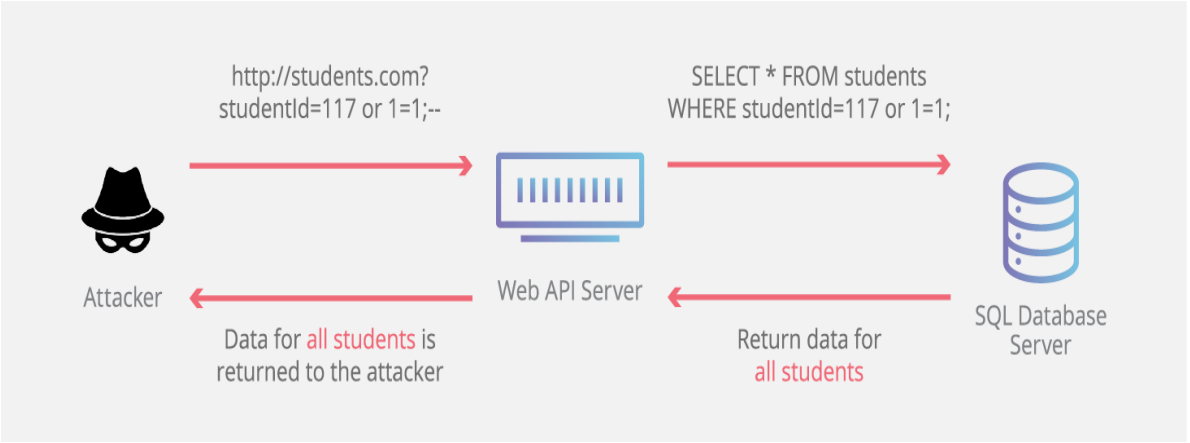

Figure 5: SQL Injection example
(Source: https://towardsdatascience.com/being-aware-of-malicious-data-corruption-as-a-data-scientist-sql-injection-attack-63f235fb2a97)

2.3 Types of Cyber Attackers

We are already familiar that attackers are individuals or groups who aim to exploit a vulnerability in a system or software and attempt to break into the system. Attackers usually have a specific goal in mind before initiating an attack.

Over time, for the purpose of the improved identification, tracking, demystifying and defending against cyber attacks, cybersecurity organizations have built classifications of different types of cyber attackers. Such segregations provide a deeper insight into the attacks which are a potential threat to the system, as well as what can be achieved in order to prevent the attacks beforehand.

An individual, an organization, or a government may build a model of a cyber criminal based on the classifications. They may use that data in order to produce a stronger analysis of what kind of attack they might expect and what would be the reason for that. By reviewing the types of attackers, the owners may prematurely define what can and what needs to be done in order to prevent intrusions while making a transparent image of the possible types of attackers who might have an intention to break into their system [1], [8].

The most common allocations of the types of cyber attackers are based on their skill level, their intention (moral ground), their relation (connectivity) to the system which is under attack by them and by their goal (gain) from intruding into the system.

2.3.1 Cyber attackers based on their skill level

Depending on the attacker's skill level, there are three different recognized skill levels: amateurs, hackers and organized hackers.

Amateurs are usually attackers with little to no skill, who take advantage of existing tools or scripts in order to initiate attacks. They usually follow instructions they find online or in a book and are known to mostly do things out of curiosity or to demonstrate their newly obtained skills. Nevertheless, if a system proves itself vulnerable enough, amateurs can still be the initiators of harmful attacks. Amateurs usually do not work well in terms of covering their tracks and can be easily traced or discovered.

Hackers are skilled and experienced cyber attackers who have the goal of breaking into a system. They are much more trained than amateurs and can recognize system or software vulnerabilities more easily. They are equipped with self-made tools and scripts which they can adapt to many kinds of systems in order to gain access. Hackers have a way of being vigilant when attacking, so as to go unnoticed while making an intrusion or not leave any digital footprint behind.

Finally there are organized hackers - or a group of hackers which may be criminals, terrorists, "hacktivists" (hacker-activists), or hackers sponsored by an organization or a state. The group of hackers are highly coordinated, trained and sophisticated and may lend their services to other criminals. These groups exist mainly in order to establish control and power and are often entangled into state politics.

2.3.2 Cyber attackers based on their intention

This classification holds the place of the most popular among all others, principally based on the entertaining nomenclature and ease of recalling. Depending on whether an attacker who attempts to break in or has already broken into a system has done it for a positive or a negative cause, the attacker can be identified as a white hat, a grey hat or a black hat hacker.

White hat hackers are ethical hackers who use their skills for a good cause and purpose. They never break into a system with a bad intention or without access. Such hackers are usually security engineers who work for a company and serve the purpose of finding issues into the system or software in order to patch it before it

has been exploited by an ill-intended hacker, for example performing network penetration tests. A penetration test is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system. On the other hand, they may be paid freelance bug bounty hunters, who charitably offer their skills to individuals or organizations in order to identify flaws in their system, so that it can be fixed before being exploited.

Grey hat hackers are individuals who commit crimes and do arguably unethical things, however not for personal gain or to cause harm. An example would be someone who breaches a network or a system without permission and then discloses the vulnerability to the owners, which gives space to the organization to repair the vulnerability without suffering prior damage. A grey hat hacker may disclose a vulnerability to the affected organization after having compromised their network, but he may also not, depending on his needs and moral standing. These kinds of attackers usually adapt the situation to their own need and agenda, so they morally stand somewhere between white and black hat hackers, thus not coincidentally the color grey in “grey hat” is deducted from is the mix of white and black color.

Black hat hackers are cyber criminals with no positive intention when breaking into a system, they violate networks, systems and programs only for malicious reasons and illegal, personal, financial or political gain. They attempt to compromise systems, cause harm or steal classified data. During the intrusion into systems, black hat hackers also are very wary of staying anonymous and not leaving any tracks which might identify them. When a hacker, cyber attacker or cyber criminal is mentioned, nearly always the initial imagined figure is a black hat hacker.

2.3.3 Cyber attackers based on their relation to the system

Depending on the attacker’s relation and connectivity to the system, cyber attackers might be identified as internal or external threats.

Internal security threats are attackers or individuals who already have access to the system and want to abuse their privilege in order to cause harm or steal data from the system. An internal threat, for example, might be an employee of a certain organization. Following from the fact that internal users have direct access to the building and its device and network infrastructure, the result of the internal attack might be a lot more significant compared to external threats. Employees also have

knowledge of the corporate network, its resources, and its confidential data, as well as different levels of user and administrative privileges. Internal attackers do not necessarily need to have hacking skills and often do not belong in any of the categories mentioned in the previous subsections, instead they solely depend on the abuse of power and privilege that they own.

External threats from amateurs or skilled attackers can exploit vulnerabilities in networks, programs or computing devices, or use social engineering to gain access. They can vary from many types of attackers which have already been mentioned in the previous subsections.

2.3.4 Cyber attackers based on their goal or gain

There are many forms of goals or benefits a certain attacker or a group of attackers might aim to achieve/obtain when attempting to break into a system or a network.
An intruder may have personal gain from the attack - such as finding out confidential data which the attacker may later use for extortion, or gain access to a social media account of a person the attacker is familiar with. There are many examples for personal gain an intruder may have and they vary depending on the individual's personality.

Similarly, if an attacker has a financial gain in mind then he/she is attempting to break into the system in order to steal financial data, such as credit card information, or virtual funds.

There exist hackers who break into a system, software or network for the sole purpose of causing damage and harm to it. They do not necessarily have a direct benefit other than self-fulfillment when performing such an offense, or they might be doing such a thing for vengeance or out of spite to somebody they know.

Organized hackers, such as hacktivists, often attempt to take control of a system in order to assert control and power and spread political propaganda to a nation. They have a specific goal and it is to ruin the public image of a government and earn the trust of the citizens.

There are terrorists who terrorise people by performing hacks on entire networks or systems within a city, a nation or a country. Such attackers have nearly identical goals with terrorists who lay terror by occupying land and carrying firearms - they do it for control, power and financial gain. These groups of attackers, as well as the organized hackers, may often serve as mercenaries to other

criminals who pay for their services, or they may be sponsored by a government in order to spread control and political propaganda.

2.4 Consequences of cyber attacks

In the past, there have been frequent cyber attacks on companies that have caused serious consequences, both financial and reputational in nature. The year 2020 has become remarkable in many ways, especially when it comes to increased frequency in cyber attacks. The growth of information systems globally has given cyber-attackers the opportunity to attack the information technology infrastructure of many organizations. Some of these attacks are data breaches or intrusions, network infiltration, major data theft and sale, identity theft and other infiltration attacks such as ransomware, malware, DDoS and phishing.

Recently, a large part of the global population moved to working from home and organizations quickly migrated to remote work. The security gap between the home and office network plays a key role in data breach in 2020. Research shows that employees working remotely (from home) have become the source of 20% of cyber security incidents, ransomware virus infiltration is on the rise, and users need to become more aware of the strength and integrity of their passwords [16]. Many companies and organizations also have yet to practice and implement a high level of security, and vulnerabilities pose a constant threat to corporate networks.

The consequences of cyber attacks may be little, sometimes unnoticeable, although they may also be devastating and cause collateral damage as well. To protect an organization from every possible cyberattack is not feasible, for a few reasons: The expertise necessary to set up and maintain the secure network can be expensive. Attackers will always continue to find new ways to target networks. Eventually, an advanced and targeted cyberattack will succeed. The priority will then be how quickly the security team can respond to the attack to minimize the loss of data, downtime, and revenue.

Consequences of a cyber attack may differ based on the type, intention, end-goal and target of the attack. We will go over a few most common consequences in the following subsections.

2.4.1 Theft

Organizations, businesses, applications and platforms which include a lot of data handling, storing and disposal are a usual target of attempted data breaches with the motive to steal, modify or wreck the information the software is handling. Smaller businesses' defenses are typically less sophisticated and easier to penetrate, making them a softer target. Cyber-enabled fraud leads to monetary losses, but stolen data can be worth far more to hackers, especially when sold on illicit and obscure markets.

2.4.2 Financial Loss

The financial impact of a cyber attack on an organization is inevitable - irregardless of the type of attack. After a breach has been detected, there are many costs which are necessary to be overcomed in order to mitigate data loss and damage to the network, system or software. The expenses which the cyber attack introduces are the cost needed to respond, investigate and repair the result of the attack. Afterwards, the introduced vulnerability in the system needs to be patched. These modes of interventions unarguably cost time as well as money. The monetary cost of a breach is much higher than just replacing any lost or stolen devices, investing in existing security and strengthening the company’s network integrity and infrastructure.

The company may be responsible for contacting all the affected customers about the breach and may have to be prepared for litigation. At this state of great disturbance, confusion and uncertainty, employees and customers may choose to abandon the company. The company may need to focus less on growing and more on repairing its reputation and finances.

2.4.3 Legal Issues and Fines

According to GDPR, there is the prospect of monetary penalties for businesses that fail to comply with data protection legislation. Global authorities are considering more difficult and strict regulations: one of the most drastic measures proposed by the European Parliament for a privacy breach, applicable from 25 May 2018, is a fine of 20 million euros, or 4% global annum revenues whichever is higher– a sum

that would threaten many growing businesses with insolvency [17]. Unfortunately for the organizations, with the enormous amount of data flow in modern applications, it is almost inevitable for a software to omit storage of data from its mode of operation, deeming data breach attacks to be much higher in occurrence than ever before.

Apart from any regulatory obligations and fines, organizations can face civil lawsuits from affected customers and business partners. If a system is breached and customer data is stolen, the organization or business may be forced to prove that the incident was not caused by negligence and that they did everything reasonably possible to maintain their best-practice security measures and procedures. Such issues further enhance the seriousness in protecting against data breach attacks.

2.4.4 Loss of Reputation

Apart from immediate costs, a cyber attack may also have less obvious long-term consequences related to reputation damage, especially if a data breach takes place. Many (if not most) organizations initially try to conceal information about attacks and breaches to minimize harm to their reputation, though this strategy can easily backfire. If the incident is exposed nevertheless, initial attempts to conceal it may increase the reputation damage and, crucially, loss of trust.

Trust is probably the most important yet most fragile aspect of any partnership or customer relationship. Customers and partners that have trusted an organization with their business and data can turn away in anger, and persuading them to stay or return will not be easy. This is especially valid in highly competitive markets with multiple stakeholders offering similar products and terms. As with any public relations crisis, a cyber attack may also tarnish the brand's image, with all associated consequences.

When a cyber attack is so severe that a business effectively grinds to a halt, the impact of lost productivity and rapidly accumulating costs may be severe enough to threaten the business continuity and foundation. In the case of organizations that depend heavily or entirely on web applications and 24/7 connectivity, cybersecurity becomes the flip side of business continuity, and just one cyber attack may be enough to put a small company out of business. With cyberattacks now considered the most likely man-made threat, cybersecurity must be a key component of any business continuity strategy.

2.5 Examples of cyber attacks with critical impacts

In this subsection we will go over a few of the latest and most prominent cyber attacks which impacted some of the largest social media platforms.
Sources: (https://theguardian.com, https://medium.com, https://theverge.com)

2.5.1 Telegram Hijack

In September 2020, an unauthorized attack on the Telegram messaging software systems and the email of clients in the cryptocurrency business was obtained through a cyber attack. Hackers used Signal System 7 (SS7), which is used to connect mobile networks around the world, to hack taxes. Signaling System 7 is an international telecommunications standard that defines how network elements in a public switched telephone network (PSTN) exchange information over a digital signaling network. According to cyber security experts, hackers have acquired the codes for the two-factor authentication (2FA).

The hackers attacked the mobile network operators who support the SMS message center in order to send a request for updating the location to at least 20 customers. The purpose of this attack was to take over different accounts and broadcast false and misleading information, as well as propaganda to many users around the world. There are improved methods for checking than just SMS or 2FA based on calls in the cybersecurity community. Cybersecurity experts believe that telecom standards must change the use of protocols such as SS7, which cannot solve modern problems.

2.5.2 Twitter social engineering attack

On July 15, 2020, social media giant Twitter had to close the Twitter accounts of some of its customers, as their profiles fell victim to a cyber-currency cyber attack. The target of this attack were public figures such as Elon Musk, Barack Obama, Jeff Bezos, Apple and Uber, who announced tweets claiming to support a charity to send funds to a cryptocurrency wallet. The tweet further claimed that the amount sent will be returned to the sender by double the amount he/she had initially sent. The investigation found that there was a gathering for a cyber attack on Twitter's servers, during which a campaign for digital fraud of over 130 profiles was launched. As it

appears, the compromise of multiple popular and influential twitter accounts was not at all a breach into Twitter's user database, instead it was a social engineering attack performed on a twitter employee who had granted permission to unauthorized users to Twitter's user administration dashboard.

2.5.3 Facebook, Instagram and LinkedIn breach

January 11, 2021: A Chinese company for the management of social media, Socialarks, underwent tax leaks through an improperly secured tax database, which revealed the details of the accounts and personal identification information (PII) of at least 214 million Facebook, Instagram and LinkedIn users. The information found for each platform varies and includes usernames, phone numbers, email addresses, profile links, profile pictures, profile listings, logistics for followers and engagements (profiling), location, job profile, relationships with other users on the social media (friends / followers).

2.5.4 Major USA East Coast pipeline shut down

As a consequence of a ransomware attack on its operating system, in May 2021, USA's East Coast largest gasoline pipeline was shut down. According to the source, the attackers had stolen nearly 100 gigabytes of data, after which they had locked the access to the system and demanded a ransom in order to release the system lock. Access to the system was restored nearly a whole day later, though it had already caused a major disruption to the fuel transport system and created panic in the nation [35].

3 OWASP FOUNDATION

As already expected, to support the ongoing effort and development of the cybersecurity sector, multiple foundations and organizations have been established to promote, spread the word, advocate, guide, and contribute to this field. Their purpose is to share the collective knowledge and experience about what we have previously learned in this thesis, as well as raise awareness to as many people as possible on this topic and make sure the internet is as safe as it could be to everyday users. Most of the organizations that support the cybersecurity field and motion are non-profit organizations and they are established and led by experts or people who have vast experience on the subject and would like to advance their knowledge onto others. The organizations are mostly voluntary, but they receive support through donations and contributions to their open-source projects, which give real-life practical examples of cyber attacks and contemporary, securely hardened applications.

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web [18]. The OWASP foundation hosts a large number of events and conferences each year, remotely and around the globe, to promote their projects, the future of the organization and to promote modern or new security standards. They own very important projects, guides and books which assist people to be involved and immersed in the latest cybersecurity trends and techniques. As a part of their projects, the foundation has various open-source frameworks, tools, proxies, and models which ease the work of cybersecurity engineers or bug bounty hunters. The projects are maintained by members of the foundation as well as voluntary members of the cybersecurity community, which are willing to lend their skills to the foundation and help by improving or further developing the active projects. Members of the community at OWASP are also able to give suggestions for new projects or guides and through them amass other interested people in order to kickstart a new project or discuss an idea.

3.1 OWASP Top 10 Web Application Security Risks

We have already gotten familiar with organizations and foundations related to web application security and how useful they are to the growth of the software and internet security branch. As we have previously mentioned OWASP (Open Web Application Security Project) and their numerous projects and guides, we should also take a look at the standards they have developed over time. The OWASP foundation contributes a great deal to developers and cybersecurity engineers by collecting quantitative and qualitative data over time through analysis and public surveys, after which the data supplies them with the latest web application security risks, metrics and tools. The most popular. prominent and the one standard with the largest influence and contribution to the digital security field is the OWASP Top 10 project.

The OWASP Top 10 is a standard awareness document for developers, security engineers and web application security. It represents a broad consensus about the most critical security risks to web applications. This project focuses on identifying the most serious web application security risks for a broad array of organizations. For each of these risks, they provide information about the likelihood and technical impact by using the following simple rating scheme, which is based on the OWASP Risk Rating Methodology.

Table 1: OWASP Risk Rating Methodology
(Source: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology)

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App. Specific	EASY: 3	WIDESPREAD: 3	EASY: 3	SEVERE: 3	App / Business Specific
	AVERAGE: 2	COMMON: 2	AVERAGE: 2	MODERATE: 2	
	DIFFICULT: 1	UNCOMMON: 1	DIFFICULT: 1	MINOR: 1	

The columns of the table show areas of applications which are observed and rated, where “threat agents” and “business impacts” are filled accordingly to the

software which is being analyzed, and the rest are where the ratings for each area are inserted The threat agents show who, why and how might attempt to hack or breach the software. The business impacts indicate the consequences that the business needs to (may) face if a cyber attack has taken place. Each other column shows the different categories of ranking with their respective rank number, where 1 is least severe and 3 is most severe.

Each organization is unique, and so are the threat factors for that organization, their goals, and the impact of any breach. If a public interest organization uses a content management system (CMS) or an information system for public information and a health system uses that same exact system for sensitive health records, the threat factors and business impacts can be very different for the same software. It is critical to understand the risk of an organization based on the applicable threat agents and business impacts [19].

In the following subsections, we will go over the current version of the project (2017) which, as the name of the project suggests, represents the top 10 web security risks according to quantitative user-submitted data for the year 2017. The listed vulnerabilities are ranked from 1 to 10 such that number 1 is the most critical and 10 is the least critical, based on the previously mentioned rating methodology. The word “critical” here should imply that if one vulnerability is ranked as more critical than another, we may assume that it can be seen as more serious, dangerous and devastating and can lead to more tangible damage and consequences if exploited by a hacker. Regarding that matter, it should be considered to address such issues and risks with higher priority and importance, giving the most effort to prevent a flaw of such kind to occur when a software becomes widely used.

3.1.1 Injection

In this paper under the common cybersecurity attacks section, we have already gotten familiar with SQL Injection as one of the most occurring and dangerous injections. Now, SQL Injection falls under the injection category of intrusions, which also includes LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages and ORM queries.

In general, as we might have already assumed from the listed attack vectors, injection happens when an attacker exploits a vulnerability that manifests through invalidated, unsanitized, unfiltered and/or unchecked user input. The fault mostly lies in the fact that the user input is directly interpolated or concatenated into

queries, commands or parsers, allowing the attacker to generate a custom query or command which is executed on the target machine whilst ignoring the rest of the hardcoded input in the program by commenting or escaping the rest of the query [7].

An easy way to test if an input field or an URL query parameter is vulnerable to injection is to test it with an input which would usually cause an error in the program. As an example, let's take the following code snippet where an SQL injection might be used:

```
"SELECT * FROM Users WHERE userId='" + insertedId + "'";
```

This code snippet represents a custom hardcoded SQL query, which presumably selects all information from a table called User, filtering the table for only the entry which has a matching "userId" data with the one the end-user has inserted or the client-side application has requested. The mistake here is that the inserted string, named as the variable *insertedId*, is directly concatenated to the custom SQL query, without previously doing any type of verification or validation on the inserted data. If an attacker has a malicious intent, he could inject a piece of SQL code that resembles the following code snippet:

```
' UNION SELECT * FROM Users; --
```

after which the resulting query matches the following:

```
"SELECT * FROM Users WHERE userId='' UNION SELECT * FROM
Users; --'";
```

If this type of query is executed, then this will not crash the program and perhaps it will not do anything like throwing an error or starting a process in the program, however, the attacker will be able to see the full response of the whole User table stored in the database.

Clearly, this type of attack is very serious and can lead to a devastating blow being dealt to the application, such as lending the attacker the ability to expose and leak the entire database schema along with its data, or perform irreversible and permanent changes such as deleting tables and records in the database or changing the permissions and denying access to already authorized users. The business impact of this type of attack depends on the type of data stored in the victim

database as well as the confidentiality of the data and the customers of the service which is under attack.

Luckily for the developers and testers of web applications who have a potential to contain such vulnerabilities, it is easy to check whether such an attack is possible by putting themselves "in the shoes" of a hacker. It is necessary to test any kind of input with the intention to break it or exploit it, in order to become aware of it and be able to fix it. It is equally important to always validate input before using it to generate a query or a command and never concatenate it directly inside a code snippet or a query.

3.1.2 Broken Authentication

Authentication is the act of validating that users are who they claim to be. This is the first step in any security validation process. Giving a person the access to a program, or any software, or providing individual users with administrative access to an application are good examples of authentication.
Authorization in a system security is the process of giving the user permission to access a specific resource or function. This term is often used interchangeably with access control or client privilege. In secure environments, authorization must always follow authentication. Users should first prove that their identities are genuine before an organization's administrators or a server grant them access to the requested resources.

Authentication is in almost all cases done through a secret phrase which every one of us recognizes as a "password", which gives the system the sign that we are the user whose account is being requested access to because we have the knowledge of the account's email and password, also known as account credentials. Other types of authentication, depending on the type and service of an application, can be in the form of one time passwords (OTP) or pins, which are usually sent to a user's email or through an SMS, such that the user inserts the received code which proves his identity. Other than that there exists authenticator apps that are connected via a user's email, whose purpose is to continuously generate new OTPs every few minutes/seconds, which grant the user access to his account (e.g. Google Authenticator), as well as biometrics (fingerprint, face scan, eye scan etc.)

Unfortunately, as time goes by, especially in the last decade, authentication methods were forced to exceed expected complexity and are not as simple as they used to be. Emails require confirmation, passwords need to be repeated, forgotten

passwords have become a process and so on. Security engineers have realized the importance of the users efficiently and genuinely proving their identity as well as securing it to a higher degree than before. This is due to the reason that broken authentication has deserved its rightful place as a very dangerous vulnerability. Broken authentication means that the authentication process for some web application was not properly and securely established and it can be easily abused by cyber attackers. It might be that the user's permissions are not appropriately or not at all verified before the user attempts to access a part of the system or perform some operation. If broken authentication is present on an application or a program, then a hacker is able to perform actions he is usually not allowed to or access other people's accounts through simple maneuvers, for example abusing a flaw in multi-factor authentication or a forgotten password mechanism.

The importance of the strength of users' passwords is today stressed as it never has been before. It is generally a very good idea to validate user passwords before allowing them to register an account, e.g. checking whether the length and content of the password fit a certain criteria. Users are asked to not use the same passwords across different accounts on different platforms and change their password at least once every month or few months.

A common authentication security measure nowadays is OAuth (open authentication), which is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites without providing them the users' passwords. It is used such that another platform (say Google, Facebook or Twitter) serves as an intermediary to prove your identity with the email you have used to register on the other platform [20]. Another secure, preferred and regularly used security measure is multi-factor authentication, which is an authentication method in which a computer user is granted access to a website or application only after successfully presenting two or more pieces of evidence to an authentication mechanism. This type of authentication adds another layer of security to a user's account, guaranteeing that the account will not necessarily be compromised only if the account's password is exposed, due to the reason that a potential intruder will also need the OTP (one-time password) in order to be granted access to the account. The most common is two-factor authentication - e.g. a user has successfully logged in with his user credentials and additionally he needs to enter a PIN or a code which has been sent to his e-mail or SMS.

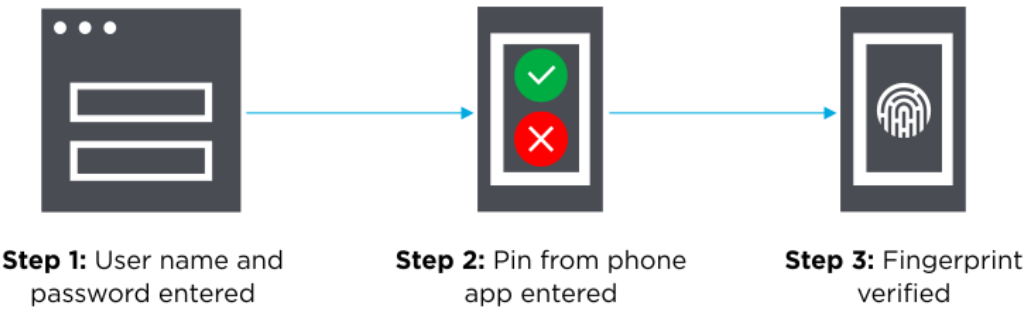

Figure 6: Multi-Factor Authentication example
(Source: https://www.onelogin.com/learn/what-is-mfa)

3.1.3 Sensitive Data Exposure

Generally, the main culprit for the exposure or unauthorized access to some sensitive, confidential and protected data is the inefficient encryption, storage and transmission of data. Data that may be identified as sensitive includes financial records and information, personal identification information (PII), health records, and other data which requires protection as defined by laws or regulations such as the EU GDPR or local privacy laws. The exposure of that type of data may lead to credit card fraud, identity theft, blackmail, abuse, or other crimes. This event not only leads to a massive disruption of the afflicted business, but it enforces it to be subjected to heavy fines due to not properly respecting privacy and customer relationship regulations.

In the current times, more than ever, companies, businesses and organizations need to put as much attention and emphasis as possible on securing the data that they frequently operate with. As far as storing passwords or other sensitive data and encryption algorithms are the matter of discussion, by today's security standards it is not in any way acceptable to store users' sensitive data as plain text in a database, however instead hashing and salting algorithms need to be used. Password hashing guards against the possibility that someone who gains unauthorized access to the database can retrieve the passwords of every user in the system. Hashing performs a one-way transformation on a password, turning the password into another string, called the hashed password. "One-way" means that it is practically impossible to go the other way - to turn the hashed password back into the original password. Several mathematically complex hashing algorithms fulfill

these needs. A cryptographic salt presents a string made up of random bits added to each password instance before its hashing. Salts create unique passwords even in the instance of two users choosing the same passwords. Salts contribute to mitigation of rainbow table attacks by forcing attackers to re-compute them using the salts.

A rainbow table is a collection of records from pre-computed password hashes and serves the purpose to attackers who want to compare hashed passwords in a database with pre-computed hashes inside the rainbow table. This type of attack is called a rainbow table attack. The danger emerges when a hacker discovers the cryptographic hashing algorithm used to encrypt the sensitive data stored in a database. After that all the attacker needs to do is use the rainbow table to crack all the data in the storage.

For this reason, unfortunately, sometimes even some older and weaker hashing algorithms are not a sufficiently sensitive data storage method. As an example, the MD5 (message digest) algorithm used to be the most frequent cryptographic hashing algorithm, though after some time numerous cryptographic flaws and weaknesses have been discovered in it and organizations or individuals who have previously used it were urged to switch to some other more secure and modern algorithms such as SHA (versions 1 and lately 2 and 3). Secure Hash Algorithms, also known as SHA are a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS) [21]. A few algorithms of interest are SHA-1, SHA-2, and SHA-3, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, similarly to MD5, is now obsolete due to the widely exposed vulnerabilities.

Besides the fact that the stored sensitive data must be properly secured and encrypted, it is also very important in what way it is transmitted - whether the communication channel is properly secured and is not vulnerable to interception, whether the sensitive data is sent as plain text or encrypted via a private and public key, etcetera. Transmitted data, also known as data in transmit, concerns protocols such as HTTP, SMTP, and FTP. Due to the increasing frequency of Man-In-The-Middle (MITM) attacks, where hackers can intercept network traffic and peek into the contents, more secure protocols, such as HTTPS, are being enforced onto websites. All data in transit is required to be encrypted with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher

prioritization by the server, and secure parameters. Encryption using directives like HTTP Strict Transport Security (HSTS) is likewise frequently enforced [22].

HTTPS stands for hypertext transfer protocol secure and is the encrypted, secure version of HTTP. It is used for secure communication across the internet or a network. The communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL). Unlike HTTP, HTTPS uses a secure certificate from a third-party vendor to secure a connection and verify that the site is legitimate. This secure certificate is known as an SSL Certificate. This is what creates a secure, encrypted connection between a browser and a server, which protects the layer of communication between the two.

This certificate encrypts a connection with a level of protection that is designated at the time of the purchase of an SSL certificate. An SSL certificate provides an extra layer of security for sensitive data that third-party attackers cannot access.

3.1.4 XML External Entities (XXE)

The abbreviation XML stands for Extensible Markup Language, which is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. XML provides a standard method to access information, making it easier for applications and devices of all kinds to use, store, transmit, and display data. XML used to be very frequently handled as a means of encoding data and enforcing data consistency, seeing the highest peak when it was mostly handled as an encoding type when sending and receiving responses from the server. However, over time that encoding has been gradually replaced by a more popular and flexible type called JSON. Hence, older systems or systems which have not been updated in a while are a common target of an XXE attack, because they have not properly configured their XML processing or documents with an XML data type are permitted to be uploaded to their application.

The XML 1.0 standard defines the structure of an XML document. The standard describes a concept called an entity, which is a storage unit of some type. There are a few different types of entities, however external general/parameter parsed entities often shortened to external entities, can access local or remote content via a declared system identifier. The system identifier is assumed to be a URI that can be dereferenced (accessed) by the XML processor when processing the entity. The XML processor then replaces occurrences of the named external entity

with the contents dereferenced by the system identifier. If the system identifier contains tainted data and the XML processor dereferences this tainted data, the XML processor may disclose confidential information normally not accessible by the application. Similar attack vectors apply the usage of external DTDs, external stylesheets, external schemas, etc. which, when included, allow similar external resource inclusion style attacks.

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

Attacks can include disclosing local files, which may contain sensitive data such as passwords or private user data, using file schemes or relative paths in the system identifier. Since the attack occurs relative to the application processing the XML document, an attacker may use this trusted application to pivot to other internal systems, possibly disclosing other internal content via http(s) requests or launching a CSRF attack to any unprotected internal services. In some situations, an XML processor library that is vulnerable to client-side memory corruption issues may be exploited by dereferencing a malicious URI, possibly allowing arbitrary code execution under the application account. Other attacks can access local resources that may not stop returning data, possibly impacting application availability if too many threads or processes are not released.

It is important to note that the application does not need to explicitly return the response to the attacker for it to be vulnerable to information disclosures. An attacker can leverage DNS information to exfiltrate data through subdomain names to a DNS server that they control [23].

Example: Retrieving the system.ini configuration file of a windows operating system by using an XXE attack, through attaching the following payload to a file and managing to upload it to a server.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///C:/Windows/system.ini" >]>
```

Through researching the cause of this attack, it is evident what developers and engineers are needed to do such that the XXE attack can be prevented or

mitigated. Any existing XML processors need to be updated, tested against the attack or rewritten. Usually it is a better practise to migrate to use of less complex data formats such as JSON, and avoid serialization of sensitive data.

3.1.5 Broken Access Control

Broken access control very closely correlates to broken authentication. Once a user has been authenticated and cleared to access a certain application, website, or another type of system, that same user is given a certain set of permissions, roles and privileges which help the system to understand what a user is able or not able to do inside the system. Reviewing a user's permissions to access a resource or perform a specific action inside an application is called authorization. If a user is authorized, that implies that the user owns the necessary permissions to access a certain resource.

Be that as it may, even though at that point it is guaranteed that the user has been authenticated under the assumption that the authentication for the system has been established properly and securely, it is still equally as important to do a security verification upon every resource request, page request or action an end user would like to accomplish. With a broken access control, hackers would be able to gain unprivileged access to hidden or restricted pages and controls on a website, for example an administration page where user activity is being monitored. Another scenario is, a hacker might obtain a privilege on a website which he/she is not authorized to by abusing the broken access control, which understands that the malicious hacker owns a certain role in a system while in reality the hacker does not. An example for this is, say, an application gives special access to a page only to users who pay for it, however a hacker abuses this and poses as a user with access to this page even though he/she did not pay for it.

Some of the most prevalent access control abusing attacks are the Cross-Site Request Forgery (CSRF) attack and the Server-Side Request Forgery (SSRF) attack. Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little assistance of social engineering (such as sending a link via email or chat), an attacker may deceive the users of a web application into executing actions of the attacker's choosing [7]. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative

account, CSRF can compromise the entire web application. Essentially, the hacker abuses the broken access control of a website, and is able to perform a malicious deed in another victimed user's name, through mimicking the user's authentication token or session cookie. An efficient way to deem an application invulnerable to a CSRF attack is by properly configuring the CORS mechanism on the server and request headers, about which we'll talk more about later in this subsection. In a Server-Side Request Forgery (SSRF) attack, the attacker can abuse functionality on the server to read or update internal resources. The attacker can supply or a modify a URL which the code running on the server will read or submit data to, and by carefully selecting the URLs, the attacker may be able to read server configuration such as metadata, connect to internal services like http enabled databases, peek into internal resources or perform requests towards internal services which are not intended to be used externally.

Simple and standard methods of authorization in recent times include JSON Web Tokens (JWT) or web cookies. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA [24]. Once a user has been authenticated, the user's claims (roles, permissions etc.) are encoded into a string of characters and digits, which can be decoded into a JSON object. Following that, the encoded token is sent as a request header with each request the user sends to the server. Given that the appropriate token has been supplied by the user, the server recognizes the user and their roles and is able to provide (or not provide) the requesting user with the requested resource. The JWTs have a limited life cycle and should always be discarded after a user has logged out or is inactive for a while. The server is always aware of JW tokens which have been issued to users who have been properly and successfully authenticated, therefore it is nearly impossible for attackers to forge these tokens. Even though JSON web tokens are considered a safe mechanism for managing access control, it is still a crucial task for the developers to properly implement the access control and verification everywhere necessary.

Another prominent mechanism for access control is CORS (Cross-Origin Resource Sharing). CORS is an HTTP-header based mechanism that allows a server to indicate any other origins (domain, scheme, or port) than its own from which a browser should permit loading of resources. CORS also relies on a mechanism by

which browsers make a "preflight" request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request [25]. For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the Fetch API follow the same-origin policy. This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers. The header that CORS uses is Access-Control-Allow-Origin, whose value is a regex string defining the allowed origins for resource sharing.

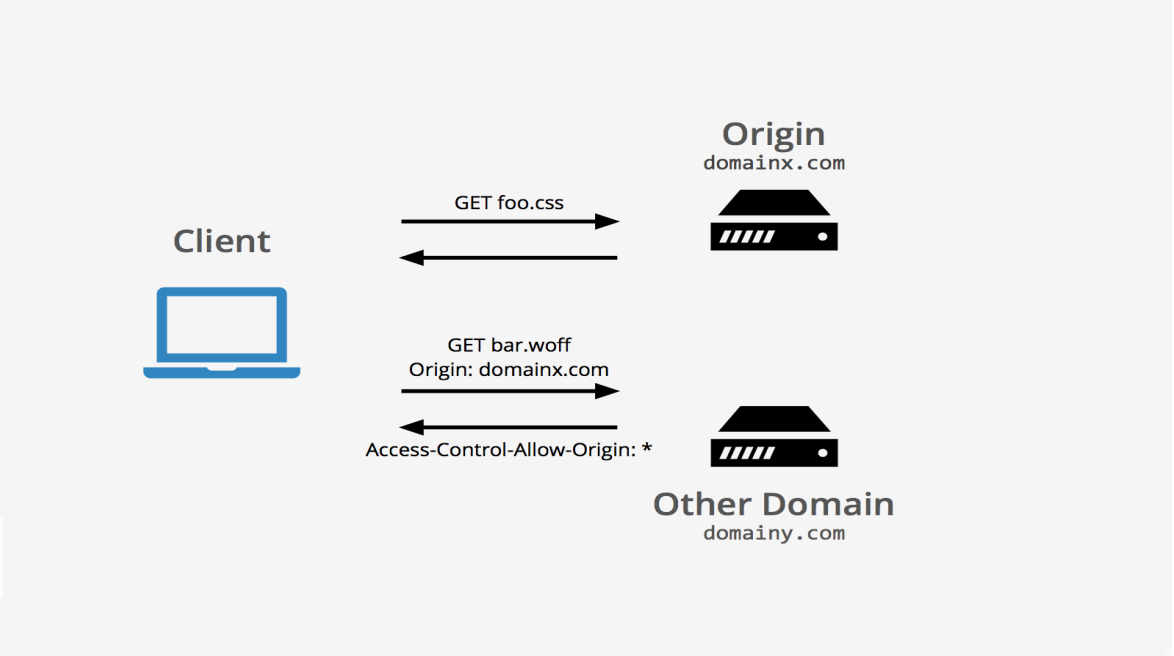

Figure 7: CORS Resource sharing visualization
(Source: https://medium.com/@buddhiv/what-is-cors-or-cross-origin-resource-sharing-eccbfacaaa30)

3.1.6 Security Misconfiguration

The security misconfiguration flaw is likely one of the most common vulnerabilities inside systems. The reason for this can have its sources in numerous variations and possibilities - one reason might be that configurations are often overlooked as something trivial, default, inherited or unimportant. Consequently, configurations are not updated and scaled in accordance with the growth of the software, its versioning as well as the versioning of its external components. Misconfigurations

may be a culprit for very large and perilous intrusions into systems, access to some of the data or functionality or even full system compromisations. If the security configurations are not properly established, a hacker might not even need a very wide skill belt in order to seize crucial information about an application [4].

Let's see an example of a misconfiguration where anyone is able to peek into the network monitoring metrics of a web application:
Let's say the web app is using Prometheus, an open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach [26]. By reading the Prometheus' documentation at, https://prometheus.io/docs/introduction/first_steps, we may notice several mentions of /metrics as the default path scraped by Prometheus, e.g. "Prometheus expects metrics to be available on targets on a path of /metrics.". Knowing this information, given the circumstance that the access to /metrics path was not securely configured, practically any person may voluntarily manually enter the path after the base URL of the web application and access the complete metrics of the entire application.
Similar scenarios include errors and exceptions which the application does not catch nor handle properly. Any request that cannot be properly handled by the server will eventually be passed to a global error handling component that sends an error page to the client that includes a stack trace and other sensitive information. The restful APIs behave in a similar way, usually passing back a JSON error object with sensitive data, such as SQL query strings or other pieces of the code. Exposing such data can reveal critical information to a potential cyber attacker, such as the type of database used, server-side related information, app-specific information and other data which might provide the attacker a lead into breaking into the system.

It is often a good idea to constantly review the configurations, preferably before every major release of the applications or after integrating a new component into the system. Unnecessary features should be disabled or uninstalled (e.g. unnecessary available ports, services, pages, accounts, or privileges), as well as making sure that a deprecated interface becomes completely obsolete and unreachable (alternatively also completely removed). Client-side or server-side errors should be handled properly as often as possible and error messages should be precise enough to let a user know what he has done wrong, though vague enough so that it cannot supply an attacker with knowledge which aids their cause. Applications should follow a segmented application architecture that provides effective, secure separation between components or modules with segmentation, containerization, or cloud security groups.

A recent project by the OWASP foundation is the secure headers project, which describes HTTP response headers that an application can use to increase its security. Once set, these HTTP response headers can restrict modern browsers from running into easily preventable vulnerabilities. The OWASP Secure Headers Project intends to raise awareness and use of these headers. The aim is also to publish reports on header usage statistics, developments and changes, code libraries that make these headers easily accessible to developers on a range of platforms, and data sets concerning the general usage of these headers.

3.1.7 Cross-Site Scripting (XSS)

Cross-Site Scripting is a type of injection attack in which an attacker is able to inject client-side javascript code from within the browser. This is a very common flaw in many web applications, therefore modern frontend frameworks and libraries usually concern themselves with preventing this vulnerability by default. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or Javascript. Escaping means that the application is programmed to mark key characters, and particularly key characters included in user input, to prevent those characters from being interpolated in a dangerous context. For example, in HTML, < can be encoded as < and > can be encoded as > in order to be interpreted and displayed as themselves in text, while within the code itself, they are used for HTML tags. If malicious content is injected into an application that escapes special characters and that malicious content uses < and > as HTML tags, those characters are nonetheless not interpreted as HTML tags by the browser if they've been correctly escaped in the application code and in this way the attempted attack is diverted.

XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface websites, or redirect the user to malicious sites. Basically the hacker is able to take control of the client side codebase and modify it according to his purpose or gain. The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, although they also commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious

operations on the user's machine under the guise of the vulnerable site. XSS flaws can be difficult to identify and remove from a web application. The best way to find flaws is to perform a security review of the code and search for all places where input from an HTTP request or user input could possibly make its way into the HTML output. Note that a variety of different HTML tags can be used to transmit malicious Javascript. If one part of a website is vulnerable, there is a high likelihood that there are other problems as well.

In newer versions of HTML, such as HTML 5, hackers have found other different ways to introduce an XSS attack into a website, such as event handlers on HTML elements. Say for example, the event handler "onload", which can be used as an attribute on HTML elements and triggers a function execution when the element has been rendered on the page's view content. Similarly to injecting the script via an input, the script may also be injected into such event handlers which trigger the malicious code execution. Other occasions where XSS is possible are in static site generators, e-mail generators which use HTML encoding to write their emails, online website builders, rich text editors, or even markdown editors.

Generally, to prevent XSS attacks from happening on a web application, there exist certain rules and guidelines which need to be followed. The main rule of XSS prevention is to treat any user input as untrusted and never embed it into scripts or HTML code, such as tag names, comments, attributes, events etc. Any string or other type of input inserted by a user on the client-side should be sanitized, properly escaped and recursively checked for script tags before being displayed or sent to the server. It is also a good idea to encode the inserted value before embedding it, if applicable.

Apart from the set guidelines designed by cybersecurity foundations and organizations such as OWASP, there also exist security mechanisms and policies which aid a web application to mitigate the possibility of XSS attacks and other types of injection or remote code executions attacks. The policies we will further mention are the Same Origin Policy (SOP) and the Content Security Policy (CSP).
Same origin policy (SOP) is a mechanism that stops one website from reading or writing data to another. It can also define what type of HTTP requests may be made to outer or inner origins. The SOP protocol analyzes three things before issuing a request - the protocol, the host and the port of the origin. If all three are the same for two different origins, then the browser allows the request to be issued. For example, the origins http://abc.com and http://xyz.com have the same protocol (HTTP),

though they do not have the same hosts (abc and xyz). If they were the same, then SOP further also checks whether the port numbers of both origins match.
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including XSS and other data injection attacks.
CSP is designed to be fully backward compatible, such that browsers that do not support it still work with servers that implement it, and vice-versa: browsers that do not support CSP simply ignore it, functioning as usual, defaulting to the standard same-origin policy for web content. CSP makes it possible for server administrators to reduce or eliminate the vectors by which an XSS attack may occur by specifying the domains that the browser should consider to be valid sources of executable scripts. A CSP compatible browser will then only execute scripts loaded in source files received from those whitelisted domains, ignoring all other scripts (including inline scripts and event-handling HTML attributes). If the site does not offer the CSP header, browsers likewise use the standard same-origin policy.

Over time, different types of XSS attacks have been discovered, with regards to the different types of vulnerabilities which led to the exploit being exposed.

3.1.7.1 Reflected XSS

Reflected XSS happens when the malicious attacker's input is directly reflected onto the user interface, most often directly as the inner HTML code of a certain DOM element. For example, a user enters a string in an input field and it is directly displayed on the web page. By abusing this functionality of the website, a hacker is able to inject a malicious script directly into the input field through embedding javascript code inside the input. Afterwards, when the DOM attempts to embed the inserted string as part of the page, the HTML gets rendered and thus the script is automatically executed by the browser. This type of XSS attack also occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request. Reflected XSS is the most common type of attack and the main culprit is no sanitization of the input parameter before displaying it on the webpage.

3.1.7.2 Stored (Persistent) XSS

This type of Cross-Site scripting attack is similar to reflected, however the input is not solely reflected back to the webpage, instead it is also stored in some database or storage before being returned by the server and displayed on the page. This may be potentially much more severe than reflected XSS, because it is actually stored in a global database and it is requested by any user who has access to the database's resources through the web application. Most regular scenarios where this type of attack takes place are forums, comment sections, visitor logs, messages, and other types of scenarios where it is common that data is being stored and also displayed to any user who chooses to view it.

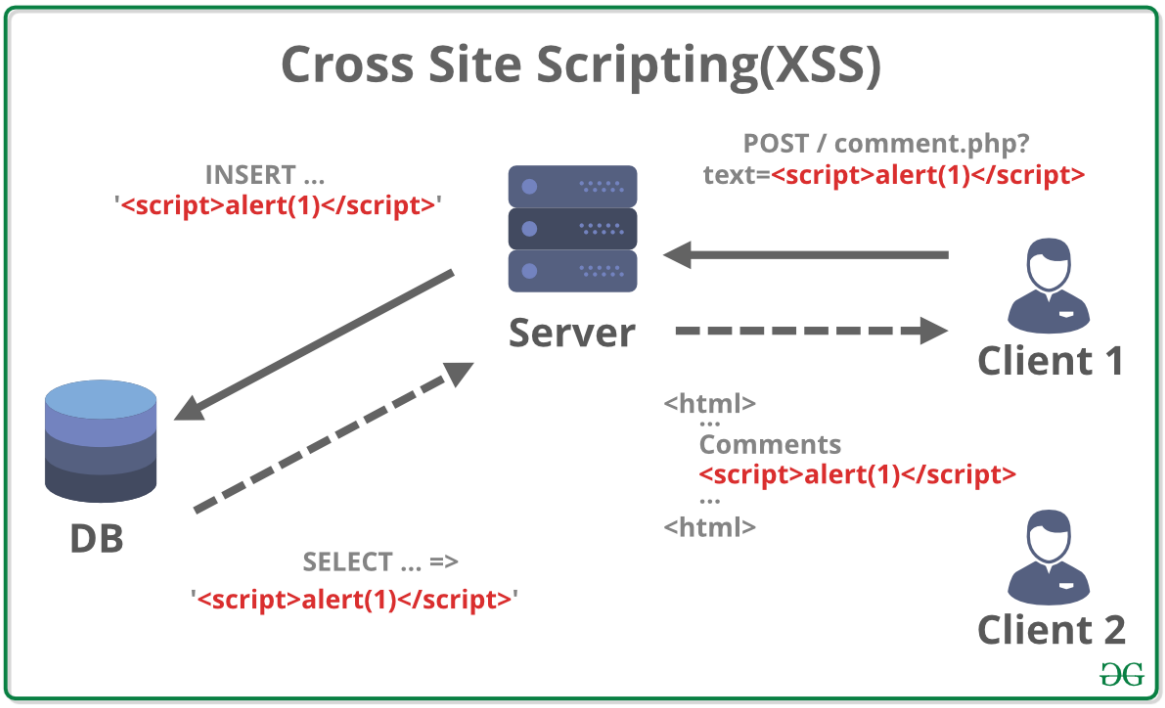

Figure 8: Stored XSS Attack Visualization
(Source: https://www.geeksforgeeks.org/what-is-cross-site-scripting-xss/)

3.1.7.3 Document Object Model (DOM) XSS

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page through which scripts can change and manipulate the document structure, style, and content. The DOM represents the document as nested nodes and objects. That way, programming languages can

connect to the page. A web page is essentially a document and this document can be either displayed in the browser window or as the HTML source, so it is the same document in both cases. The Document Object Model (DOM) represents that same document so it can be manipulated and it is an object-oriented representation of the web page, which can be modified with a scripting language such as Javascript. [27]

When DOM XSS is in question, the user's input directly lands inside a perilous part of the client-side javascript code. For example, HTML elements contain the innerHTML attribute. Inside this attribute we are able to insert any HTML code which is then rendered as an inner snippet of the element. However, if a user input is directly embedded inside the innerHTML attribute of an HTML element, then this is a direct flaw which injects the input inside the DOM of the website and can lead to an XSS attack, because an attacker can use the <script> HTML element in order to embed javascript code scripts which are then immediately executed.

Further reading: [28]

3.1.8 Insecure Deserialization

Serialization is the process of transforming some object into a data format that can be restored later to its original state. People often serialize objects in order to save them to storage, or to send as part of communications. Deserialization is the reverse of that process - taking data structured from some format, and rebuilding it into an object. Today, the most popular data format for serializing data is JSON. Before that, it was XML. The need for serialization comes from the fact that a server is only able to receive, intercept and understand only certain types of data, or the need arises to send our data over the network or have it stored in files. Without serialization, the problem might be noticed in the network infrastructure, or the hard disk is a hardware component that understands bits and bytes though not programming objects. Nevertheless, in these cases there exists the need for raw data to be serialized.

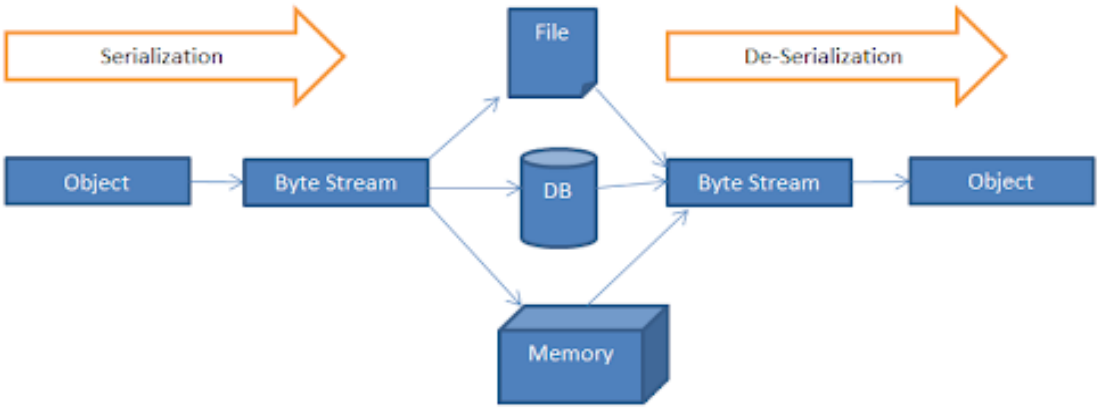

Figure 9: Serialization and Deserialization visualization
(Source: http://mrbool.com/serializing-and-de-serializing-objects-in-java/36783)

Many programming languages offer a native capability for serializing objects. These native formats usually offer more features than JSON or XML, including customizability of the serialization process. Regrettably, the features of these native deserialization mechanisms can be repurposed for malicious effect when operating on untrusted data. The main problem is that arbitrary and untrusted code is being deserialized. Hence, a hacker is able to input some malicious code snippet inside the code which is supposed to be deserialized. Upon serialization or deserialization on a set of data, the code may reveal a large problem which has been hidden inside the object or data and perform a code execution on the system or cause the system to crash. The impact of deserialization flaws cannot be overstated - these flaws can lead to remote code execution attacks, one of the most serious attacks possible.

Remote code execution is a type of vulnerability where an attacker is able to run code or scripts of their choosing with system/administrator level privileges on a server that possesses the appropriate weakness. Once sufficiently compromised, the attacker may indeed be able to access any and all information on a server such as databases containing information that unsuspecting clients provided and other confidential data. Also the attacker is able to find other system information which might aid him to further increase the harm done to the server and the person or the entire business who owns it. What makes this particularly problematic is not only the real threat of information theft and other risks associated with running arbitrary code on the server, but the difficulty in detecting this defect. Uncovering this defect

may be challenging at best and impossible at worst. Extensive code and other reviews of the web application may be impractical if not impossible. Later penetration testing may assist in discovering these defects and should be done in the case of more sensitive applications that handle sensitive information.

Let's see an example of a case of insecure deserialization which leads to remote code execution: We have an e-commerce system which lets us make purchases of items through their website. We can deduce that for each order settled, a POST request is being issued to the server, with the JSON formatted body:

```
[{"productId": 5,"quantity": 3,"customerId": 1076 ,"orderId": "sp2m4qkz"]
```

Now, a vigilant hacker might observe this behaviour, and try to manually issue a POST request to the server by using an API mocking tool. The body of the POST request is refactored as such:

```
[{"productId": (function dos() { while(true); })(),"quantity":
3,"customerId": 1076 ,"orderId": "sp2mqxkz"]
```

An insecure JSON deserialization would execute any function call defined within the JSON String, so the payload is an endless loop which initiates a denial of service on the server. By replacing the productId value with a javascript immediately-invoked function expression (IIFE), the code is executed remotely on the server upon deserialization of the data.

A different scenario where this attack can happen which does not involve sending data to the server are the website's cookies. An HTTP cookie (also called web cookie, internet cookie, browser cookie, or simply cookie) is a small piece of data stored on the user's computer by the web browser while browsing a website. Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user's browsing activity (including clicking particular buttons, logging in, or recording which pages were visited in the past). They can also be used to remember pieces of information that the user previously entered into form fields, such as names, addresses, passwords, and payment card numbers. Cookies are stored as a Base64 encoding of textual or JSON data. For the application to see the contents of the cookie, it needs to decode (deserialize) the data. A hacker is here able to hijack the stored cookie and dump some malicious code in its contents, such that the code gets executed when the cookie gets deserialized, similarly as in the previous example.

By default, comparably to other attacks such as XSS and Injection, the arbitrary input an application or a server obtains should be untrusted. Implementing integrity verifications such as digital signatures on any serialized objects to prevent hostile object creation or data tampering is always preferred before deserializing the data.

3.1.9 Using Components with Known Vulnerabilities

Given today's software application development standards, requirements and methodologies, especially when considering web or mobile applications, it is almost unavoidable for a framework, library, or other type of importable module to be used during software development. Such additions to the application decrease the development time and complexity and provide the developers with prebuilt reusable and customizable software. It is proven that the proper use of libraries increases customer and developer satisfaction, as much as it decreases the need for developing everything the software specifications demand from the beginning.

Despite all the easiness, satisfaction and benefits additions such as frameworks, libraries and packages provide, it should not be forgotten that after all they are likewise software, and software is prone to containing undiscovered vulnerabilities. An important factor here is that all of the additions to the software in the form of packages, modules, or anything similar, function in the same privilege level as the software they are contained in. With respect to that fact, if a web application is using a component which contains a vulnerability, a hacker might exploit this weakness and intrude in the application, disregarding the fact that the application may already be very secure and hardened opposed to cyber attacks. While some known vulnerabilities lead to only minor impacts, some of the largest breaches to date have relied on exploiting known vulnerabilities in components. An example of a known vulnerable software which had serious impacts after being discovered and exploited by hackers is the sanitize-html hosted on the node package manager repository. It was shown that the sanitization of HTML strings is not applied recursively to input, allowing an attacker to potentially inject script and other markup [29]. This implies that if a hacker would input a malicious script, for example:

```
<script>javascript:alert('XSS')></script>
```

Then this script would be sanitized and not executed. However, the cunning hacker would then rewrite the injected script as something similar to

```
<<script></script>iframe src="javascript:alert('XSS')">
```

and would successfully perform an XSS attack because the input is not being recursively sanitized, but only escapes the element tags on the highest level.

Hackers in the past have been able to abuse such flaws even further by developing malicious components themselves and distributing them as legit, genuine and helpful components. When they detect that their component has been installed somewhere, they are able to use this to break into the web application. Another common abuse that hackers have been known to use is called typosquatting. Typosquatting is a form of cybersquatting, and possibly brandjacking which relies on mistakes, specifically typos made by unknowing users downloading a package or a library from a popular registry to their software. If a developer accidentally makes a typo and enters an incorrect package name, say by a means of swapping two letters, they may download a component which seemingly does what it is supposed to, although behind the hood it is a copy of the original component, spiced with an intentional vulnerability so a hacker is able to abuse that to break into the victimed application. Example: The original component's name is python-tester, while a hacker makes a copy of this component and names it python-tseter intentionally. He is then reliant on some developers making a mistake while typing and installing what appears to be a genuine component.

This type of common vulnerability lets us know that it is very important that developers remain aware of the components they add to the software they are developing. The installation needs to be double-checked, the versions need to be tracked and regularly updated and they should always be installed from official, legit and secure sources. In recent times there have been creative ways that this vulnerability is being battled. There exist automatic dependency analyzers which warn developers if they are using an outdated version of a component or there is an update available. They can even be integrated with version control platforms and CI/CD pipelines, making sure that the application compilation and build does not succeed until all dependency needs of the software are met. Node package manager (NPM) is today the largest repository of software packages, libraries and components built by other developers and it is composed of completely open source components which are hosted on GitHub. This provides a total transparency of the software which supports the component and makes it secure and reliant, as well as enabling community members to fix flaws that they potentially detect while using the free software.

3.1.10 Insufficient Logging & Monitoring

In computing, a log file is a file that records either events that occur in a software system or other software runs, triggers, user actions and network requests. Any action a user performs in the system, any event which is triggered by an action, any server response message and other relevant details are kept inside the log. Logging is the process of keeping a log. Many operating systems, software frameworks and programs include a logging system. Monitoring a software is similar to logging, except that monitoring is used to track ongoing activity inside a system as well as collect metrics and usage statistics, which are used to have an improved overview of the system requirements and system management.

Logging and monitoring are often overlooked and deemed as unnecessary. Although they do not directly pose a flaw or a threat to the application's security, their absence and non-utilization may provide cyber attackers with the freedom to probe the application for flaws without the developers, system and network engineers noticing. Most successful attacks commence with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploits to nearly 100%. [30] It is also important to note that logging and monitoring should be implemented properly, such that it has a centralized and secure area of storage. The importance comes with the fact that if a hacker would access the server/system logs, then he would be provided with critical information which cannot be disclosed to anyone except the engineers, allowing the hacker to gain a deeper knowledge about how the system works internally and giving him a big lead towards breaking into the system.

To properly generate logs, they need to be logged in a readable and easily consumable format. It needs to be ensured that all login, access control failures, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious activity, and held for sufficient time to allow forensic analysis. The logs also need to be traceable such that they provide the exact source of the action/event, as well as the time of execution and the user who had issued the request.

3.2 OWASP Juice Shop

The OWASP Juice Shop is another ambitious project hosted by the OWASP foundation, which represents an online web application that is intentionally critically insecure and is developed and maintained by volunteers.
According to the organization, "OWASP Juice Shop is probably the most modern and sophisticated insecure web application! It can be used in security training, awareness demos, CTFs and as a guinea pig for security tools. Juice Shop encompasses vulnerabilities from the entire OWASP Top Ten along with many other security flaws found in real-world applications". The OWASP Juice Shop is supposed to be the complete opposite of a best practice or template application for web developers - it is rather an awareness, training, demonstration and exercise tool for security risks in modern web applications. The project comes together with a book written by Björn Kimminich, who is the lead developer and cybersecurity engineer in the OWASP Juice Shop project [2], [31].

"*Translating "dump" or "useless outfit" into German yields "Saftladen" which can be reverse-translated word by word into "juice shop". Hence the project name. That the initials "JS" match with those of "JavaScript" was purely coincidental!* " *[31]*

4 EXPOSING A VULNERABILITY

As a contribution to this thesis and the research in the field of cybersecurity, as well as software security and common vulnerabilities, I have chosen to give a practical example of a software vulnerability, how it is exploited and how threatening it can possibly be. This contribution will demonstrate how I have explored, probed and eventually detected a vulnerability in a publicly available software. We will also take a look at how the previous knowledge contained in this thesis helped to observe and determine this flaw, as well as to determine the category of this vulnerability based on the OWASP top 10 tier list, its potential business impact and the consequences.

The software vulnerability which I will further explain in the following chapters has already been privately disclosed to the owners of the software and has in no way been abused or shared elsewhere. By the time of presenting this thesis, the vulnerability is already patched and can no longer be exploited.

4.1 Software description

The software we will be inspecting is a website which goes by the name of KlikniJadi (transl. Click and Eat) [32]. It is a very popular and verified food delivery company which offers a great variety of food choices and delivery types. The business has a close relation and partnership with a vast majority of restaurants and food chains across the country, therefore offers many restaurants and food places to choose from. KlikniJadi is considered to be the most popular and most profitable food delivery service in the country. They are likewise owners of the largest food delivery chain in the country. Consequently, as food delivery is becoming more and more utilized, many new users are joining daily and are often using KlikniJadi's public website to order food.

This already becomes a solid reason for the team behind the project to be more dedicated to security, validation and proper authentication and authorization. Despite its straight-forwardness and mainstreaming, it is still important to consider that users are required to insert private information on the platform, due to the fact that such information is obligatory for food delivery to be possible.

4.2 Exposed security vulnerability

According to Juice Shop [31], before investigating an application for security vulnerabilities, we should never blindly start throwing attack payloads at it. Instead, we should make sure that we understand how the application works internally before attempting anything. This includes the application structure, its set of components and features, its underlying technologies, etc.

Generally speaking, application developers tend to follow the same project development structure when building new features or components, and pieces of the software follow pre-established practises and conventions. This maximizes reusability and project organization, while facilitating the development and decreasing the development time. This is very easy to notice from a developer's point of view. This approach is widely acceptable and even recommended, however from a security point of view, it may expose the application to certain risks, following from the fact that the development approach of reusability may spread a common flaw across many new features or components. As an example, if the authorization of a certain application endpoint is not properly configured, and it is later reused to another page/component, then the new component will also contain the equivalent flaw.

Based on my observation, this has been the case for KlikniJadi. Another thing worth noting is that the developers of this software have been "walking the happy path". What is known as the "happy path" in software engineering and software testing is an approach which does not take into consideration some exceptional scenarios and/or edge cases [33]. During the development in this manner, the developers only consider the default, expected scenarios and picture their users as generic and stereotypical for that type of service. This leaves out a lot of space for software vulnerabilities to appear, as the validation, authentication and authorization are most likely done only superficially and sufficiently enough only for a person who would never use the application in an unexpected manner. Errors are not handled gracefully or are not handled at all, leading from the fact that they were not previously deemed plausible. An example I can give specifically for KlikniJadi is, many of the validations are done only on the client's side of the application. For example, when creating an order, a customer is required to insert the date and time for the delivery. The customer is then required to insert this in a custom date and time picker interface, where he/she is only able to choose from the

available time slots proposed by the application. The input field already has a built-in validation method and does not allow inserts of unavailable timeslots or another string which is an invalid format. However, it has not been taken into account that somebody may attempt to do this via a custom server request other than through the service's website. If a custom request is made to the server and an invalid date format or delivery time is inserted, then this is accepted by the server due to insufficient validation. If this were to be exploited, then an abusive attacker may create a lot of similar requests and populate the database with orders at invalid timeslots, which may be the cause of a heavy and dangerous business impact.

A final security observation to make about the food delivery service provider is that the underlying technology of the website is publicly listed in the developer's portfolio. It is understandable that the developers had an intent to publicly showcase their skills, knowledge of a certain programming language, framework, or tool, however this is very useful information in the eyes of a cyberattacker. The hacker may abuse this knowledge to find vulnerabilities in the software that may not even be the developers' mistake. There exist numerous hacker databases and forums where vulnerabilities are being exchanged on a daily basis. A hacker may determine a flaw related to a specific framework, tool or programming language which may lead him to exposing that very flaw on a software which is built by using that technology.

All of the previously discussed topics have led me to detect a severe security vulnerability located on KlikniJadi's profile page. If abused, the vulnerability is able to open a door to populating their database with fake accounts or blocking existing users from being able to modify their own profiles. Next, we will see how this is possible. On the profile page we are presented with our current profile information and input fields to modify our profile if we wish to. The username and e-mail fields are read-only and can not be modified. After inspecting the network requests in the developer tools, we see that upon saving changes made to the profile, an HTTP POST request is sent out to the '/MyProfile' server endpoint, and all of the profile data is being sent in the oblique of a form-data object. The Form Data interface provides a way to easily construct a set of key/value pairs representing form fields and their values [34]. The form data is always associated with a <form> HTML element wrapped around all of the input fields. This provides an easily accessible interface to all inputs on the page, including the form action upon submission and the necessary input validators. When inspecting the form data object that is being sent to the server upon profile modification, we can also notice a few other values

being sent, such as Id and other user data which is not meant to be disclosed to the user.

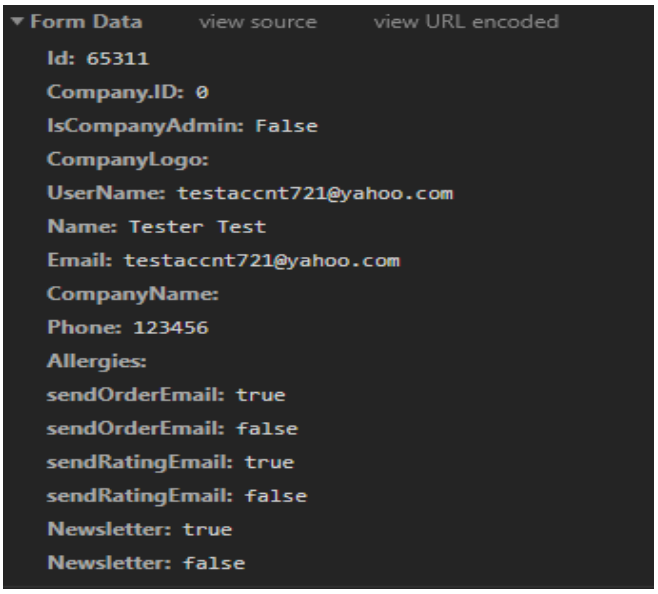

Figure 10: Profile modification form data example
(Source: kliknijadi.mk profile page, Google Chrome developer tools screenshot)

Coming from the fact that we are on the user profile page, we may assume that the Id is the logged in user's unique identification number, used such that the server may understand which user had requested the modification of their profile page. Since the data sent is a form data object, we can know for sure that the Id input field exists somewhere on the page and it is most likely hidden. We can easily prove this to be true through the developer tools, where we can find and show the field on the page, as well as verify that it is automatically filled with the current user's Id.

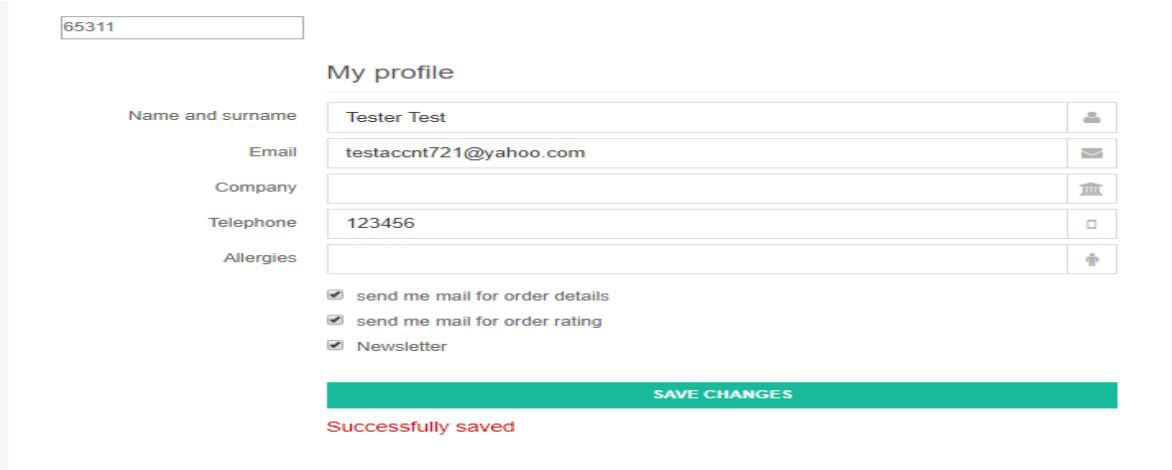

Figure 11: Profile page and visible user ID Field
(Source: kliknijadi.mk profile page)

The Id input element embedded in the page's HTML code looks as following:

```
<input data-val="true" data-val-number="The field Id must be a
number." data-val-required="The Id field is required." id="Id"
name="Id" type="hidden" value="65311">
```

It is evident that we can remove the "type='hidden'" attribute in order to show the input field on the page. If we then modify the Id to an arbitrary number and save some changes, then they are not applied to our profile page. This way we immediately know that they are either being applied to another user's profile or the request is ignored, however we do not receive any kind of validation error, which implies that such a request has not been properly handled.

For this purpose I have created the test account in the previous figure, apart from my original email account. After testing that scenario, it is shown that it is not possible to modify another account's information through this technique and the reason for that is the authorization token is parallely being sent to the server. If there is a workaround for this or if a hacker is able to steal or forge a request token, then this would be possible and would be enormously dangerous and devastating to the business. Though there is still a vulnerability which may cause a heavy impact - and that is disabling a user's privilege to be able to modify their own account. This can be done through making an account modification request with an arbitrary Id (most likely a not-yet existing in the database) and another user's email address. It is again possible to modify the email input field through the developer tools. I have made a request from the test account with an arbitrary large number and the e-mail address from my other "original" account. When performing the request, nothing is changed on the page and we do not get any feedback from the server. However, when browsing back to the profile page from the other account and trying to make a change, we get back the following message from the server, which says "User with the same email already exists". A screenshot of the page displaying the error message and the discussed form is visible on the next page.

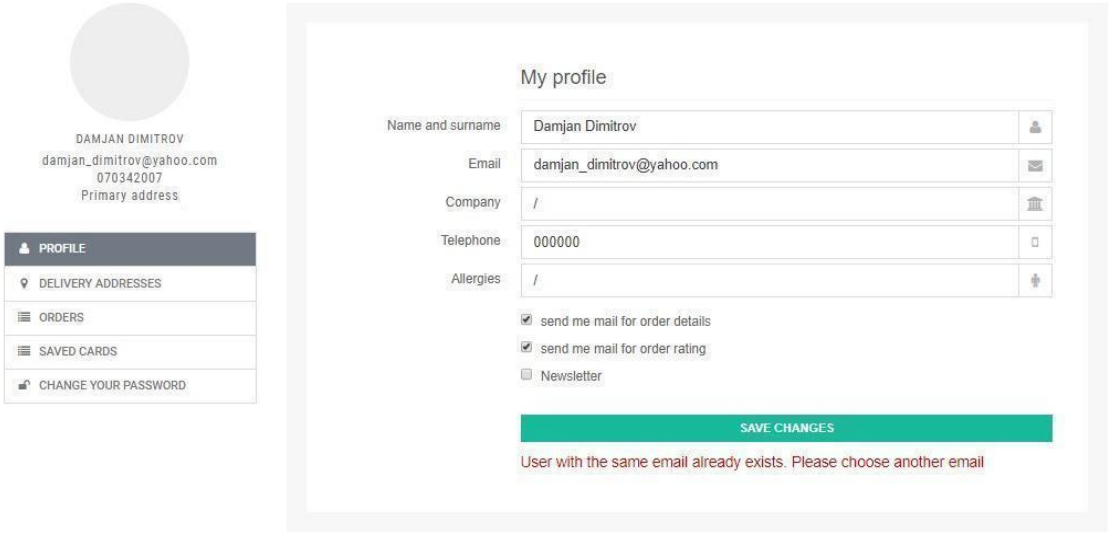

Figure 12: Profile modification error message
(Source: kliknijadi.mk profile page)

I have come to a conclusion that there is no way to "go around" this issue and I have essentially lost the ability to modify my profile information. This is a quite inadequate error message when a person attempts to modify the own profile information, so we can assume that it comes from the registration action on the server, perhaps modifying a user's information implies that a user is "re-registered" with the same email, username and Id, though the modified information overwrites the previously stored information.

This vulnerability may cause a very large impact on the business, namely a cyberattacker with a bad intention may purposely perform the equivalent attack payload to many other user accounts. Such problems stem from the fact that we have already mentioned - and that is the fact that there is no appropriate input validation. The validation is only done on the client's side, though when some data is sent to the server it is not re-validated, for example - does the email address sent match the current user's email (likewise for the user's Id). Consequently, this software vulnerability belongs in the category of "improper input validation".

5 CONCLUSION

In this thesis we have taken an overview of some aspects of the cybersecurity field. We have gotten familiar with what cybersecurity is and what it stands for. The thesis was mostly oriented towards inspecting, testing and acknowledging various different categories of cyber attacks, software vulnerabilities and malicious software. The vulnerabilities we have gotten through are only a surface-view among the vast amount of other possible software vulnerabilities and vulnerability types.

Over time, the field of cybersecurity is becoming more prominent, widespread and acknowledged, as many people are becoming more aware of common security and privacy issues and risks. Websites are being built in a security- first, features-later approach, multi-factor authentication is openly encouraged by services where private information is stored, many new organizations and foundations focused on software security appear and the demand for cybersecurity engineers grows larger over the years.

Many corporations around the world have already come to terms with the fact that the security of their systems, networks and software are no longer something which is good to have, but it is an obligation at the least. Consequently, there is much higher demand for experts or skilled people who are able to contribute to the security strength of a business or a company. Many new kinds of employment opportunities and job offerings have come to life as well. Interestingly enough, there are people who are able to offer their hacking skills as freelance work, where they are paid to find security flaws in a company’s system or software. They are white-hat hackers who are given permission to execute attack payloads at applications and systems in order to detect vulnerabilities and later report it to the owners, after which they are being paid for their work. Such hackers are most often called “ethical hackers”.

By this point it is already evident how significant, imperative and necessary software security is and how paramount the field of cybersecurity is to the domain of computer science, software development, systems and network engineering, as well as to other fields. The cybersecurity field continues to rapidly evolve and adapt with the goal of spreading awareness to a larger number of internet users and continuously establishing new security principles and standards.

6 POVZETEK NALOGE V SLOVENSKEM JEZIKU

V zaključni nalogi se ukvarjamo s pojmom kibernetske varnosti, ki je znana tudi kot računalniška varnost, varnost programske opreme ali varnost informacijske tehnologije. Gre za skupek metod, ki se ukvarjajo z zaščito različnih digitalnih komponent, kot so podatki, aplikacije, omrežja in računalniški sistemi, pred nepooblaščenim dostopom, krajo ali poškodbami. Ranljivost programske opreme predstavlja napako ali slabost pri načrtovanju, izvedbi ali funkcionalnosti določene komponente, ki pripada tej programski opremi. Ranljivost je pogosto mogoče uporabiti kot vstopno točko za napadalce, da bi lahko zlorabili programsko opremo, na primer krajo informacij ali pridobitev privilegiranega dostopa do drugih komponent. Ranljivosti se lahko pojavijo ob številnih priložnostih - v sami programski opremi, računalniških omrežjih, ali programskih knjižnicah drugih razvijalcev, ki jih je mogoče preiskati, predelati, ali izkoristiti z uporabo več napadalnih tehnik, skript in orodij. Kibernetska varnost se ne nanaša samo na ranljive dele programske opreme, pač pa naslavlja tudi vzpostavitev številnih pravil, standardov, načel, praks in politik, ki jih je treba upoštevati pri načrtovanju in razvoju informacijskih sistemov.

V nadaljevanju se posvetimo najpogostejšim vrstam napadov programske opreme in metodam, ki jih kibernetski napadalci uporabljajo za vdor v programsko opremo. Poleg ranljivosti si ogledamo tudi identitetni profil kibernetskih napadalcev ter posledice njihovih napadov. Pokažemo tudi nekaj primerov kibernetskih napadov, ki so se zgodili leta 2020, vključno z načini napada in posledicami.

K izboljšanju kibernetske varnosti veliko prispeva neprofitna fundacija OWASP. V nalogi povzemamo njihov projekt »OWASP Top 10«, ki predstavlja široko soglasje o najbolj kritičnih varnostnih tveganjih za spletne aplikacije in s tem standardni dokument ozaveščenosti za razvijalce in varnostne inženirje. Osredotoča se na prepoznavanje najresnejših varnostnih tveganj spletnih aplikacij ter vsako od teh tveganj zagotavlja informacije o verjetnosti in tehničnem ter organizacijskem vplivu ranljivosti.

Nalogo zaokroža praktičen primer ranljivosti javno uporabljene spletne aplikacije. Prikazan je postopek identifikacije ranljivosti, način možnega napada in možne posledice v primeru zlorabe. Ugotovljeno ranljivost smo prijavili lastnikom in razvijalcem, ki so jo do objave zaključne naloge že odpravili.

7 REFERENCES

[1] Beck Buchanan, "The Intruder's View" in *The Cybersecurity Dilemma: Hacking, Trust and Fear Between Nations*, Oxford University Press; 1st edition, 2016

[2] Charles Arthur, *Cyber Wars: Hacks that Shocked the Business World*. London, England: Kogan Page, 2018

[3] Andrew Honig and Michael Sikorski, *Practical Malware Analysis: The Hands-on Guide to Dissecting Malicious Software.* San Francisco, CA: No Starch Press, 2012

[4] Adam Shostack, "Strategies for Threat Modeling" in *Threat Modeling: Designing for Security.* Hoboken, New Jersey: Wiley, 2014

[5] Mary Aiken, *The Cyber Effect: A Pioneering Cyberpsychologist Explains How Human Behavior Changes Online*. New York: Random House; Reprint edition, 2016

[6] Christopher J. Hadnagy, "Psychological Principles Used in Social Engineering" in *Social Engineering: The Art of Human Hacking*. Hoboken, New Jersey: Wiley, 2010

[7] Björn Kimminich, *Juice Shop*. Canada: Leanpub, 2021

[8] Cisco Systems,"Cisco Networking Academy - Introduction to *Cybersecurity" netacad.com,* Feb. 2018. [Online]. Available*:* https://netacad.com/courses/cybersecurity/introductions-cybersecurity. [Accessed Mar. 2021]

[9] Wikipedia, free encyclopedia, "Computer Virus*"*, *wikipedia.com,* Feb. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Computer_virus. [Accessed Feb. 2021]

[10] Kaspersky Internet Security, "What is a Trojan Virus", *kaspersky.com*, Dec. 2019. [Online]. Available: https://www.kaspersky.com/resource-center/threats/trojans. [Accessed Feb. 2021]

[11] CSO Online, "What is a computer worm, how this self spreading malware wreaks havoc", Aug. 2019. [Online]. Available:

https://www.csoonline.com/article/3429569/what-is-a-computer-worm-how-this-self-spreading-malware-wreaks-havoc.html. [Accessed Feb. 2021]

[12] Phishing.org, "Phishing, General Phishing Information and Prevention Tips", Apr. 2017. [Online]. Available: https://www.phishing.org/. [Accessed Feb. 2021]

[13] Christian Ludeman, "The Complete Guide to Angular Security", Mar. 2021. [Online]. Available: https://christianlydemann.com/the-complete-guide-to-angular-security/. [Accessed May 2021]

[14] Zscaler.com, "Ubiquitous SEO Poisoning URLs", Oct 2018. [Online]. Available: https://www.zscaler.com/blogs/security-research/ubiquitous-seo-poisoning-urls-0. [Accessed May 2021]

[15] Computerphile, "DNS Cache Poisoning", *youtube.com* Jul 2020. [Online]. Available: https://www.youtube.com/watch?v=7MT1F0O3_Yw. [Accessed Apr. 2021]

[16] ZD.net, "Working from home causes surge in security breaches, staff 'oblivious' to best practises", Aug. 2020. [Online]. Available: https://www.zdnet.com/article/working-from-home-trend-causes-surge-in-cybersecurity-costs-security-breaches/. [Accessed May 2021]

[17] General Data Protection Regulation (GDPR), "GDPR Fines and Penalties", *gdpr-info.eu,* 2018. [Online]. Available: gdpr-info.eu/issues/fines-penalties/. [Accessed Apr. 2021]

[18] OWASP Foundation, "Open Source Foundation for Application Security", 2001. [Online]. Available: https://owasp.org/. [Accessed July 2021]

[19] OWASP Foundation, "OWASP Top Ten Web Application Security Risks", *owasp.org*. [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed Jul. 2021]

[20] Open Authentication Protocol (OAuth), "OAuth Community Site", Apr. 2010. [Online]. Available: https://oauth.net/. [Accessed Jun. 2021]

[21] Wikipedia free encyclopedia, "Secure Hash Algorithms", *wikipedia.org,* July 2021. [Online]. Available: https://en.wikipedia.org/wiki/Secure_Hash_Algorithms. [Accessed Jul. 2021]

[22] Scott Helme, *"*HSTS - The missing link in Transport Layer Security*"*, Nov. 2013. [Online]. Available: https://scotthelme.co.uk/hsts-the-missing-link-in-tls/. [Accessed Jul. 2021]

[23] OWASP Foundation, "XML External Entity (XXE) Processing", *owasp.org,* Dec. 2021. [Online]. Available: https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing. [Accessed May 2021]

[24] JavaScript Object Notation Web Tokens standard, "JSON Web Tokens", 2010. [Online]. Available: https://jwt.io/. [Accessed Mar. 2021].

[25] Mozilla Developer Network (MDN), "Cross-Origin Resource Sharing (CORS)*", developer.mozilla.org,* Jul. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Accessed Jul. 2021]

[26] Soundcloud, "Prometheus - Monitoring system & time series database*",* *soundcloud.com*, 2012. [Online]. Available: https://prometheus.io/. [Accessed Mar. 2021]

[27] Mozilla Developer Network (MDN), "Introduction to the DOM, Web API", *mozilla.org,* Jul. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. [Accessed Jul. 2021]

[28] OWASP Foundation, "Cross Site Scripting (XSS) Software Attack", *owasp.org*, Dec 2017. [Online]. Available: https://owasp.org/www-community/attacks/xss/. [Accessed Apr. 2021]

[29] Snyk.io, "Cross-site Scripting (XSS) in sanitize-html", Aug 2016. [Online]. Available: https://snyk.io/vuln/npm:sanitize-html:20160801. [Accessed May 2021]

[30] OWASP Foundation, "A10:2017-Insufficient Logging & Monitoring", Nov 2017. [Online]. Available: https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging %2526Monitoring. [Accessed Apr. 2021]

[31] OWASP Foundation and Björn Kimminich, "Juice Shop - Insecure Web Application for Training", *owasp.org,* Dec. 2016. [Online]. Available: https://owasp.org/www-project-juice-shop/. [Accessed Jul. 2021]

[32] KlikniJadi, "Food Delivery Service - KlikniJadi (Click and Eat)", 2021. [Online]. Available: https://www.kliknijadi.mk. [Accessed Jul. 2021]

[33] Wikipedia, free encyclopedia, "Happy Path", *wikipedia.org,* Feb. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Happy_path. [Accessed May 2021]

[34] Mozilla Developer Network (MDN), "Form Data", *mozilla.org,* Jun. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/FormData. [Accessed Jun. 2021]

[35] Nymag.com, "Colonial Pipeline Shutdown", Jun 2021. [Online]. Available: https://nymag.com/intelligencer/article/what-we-know-about-the-colonial-pipeline-shutdown-updates.html. [Accessed Jun. 2021]