



LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

The web-based vulnerabilities, stakeholders and avoidance

Siltainsuu, Janne

2018 Leppävaara

Laurea University of Applied Sciences
Leppävaara



The web-based vulnerabilities, stakeholders and avoidance

Janne Siltainsuu
Security management
Bachelor's Thesis
July 2018

Janne Siltainsuu

The web-based vulnerabilities, stakeholders and avoidance

Year	2018	Pages	29
------	------	-------	----

This bachelor's thesis is focused in cyber security and the web-based vulnerabilities and stakeholders who participate in the exploitation as being the technical equipment, the attacker or the victim. The focus of this thesis is to first give the reader the picture about the cyber world and its stakeholders and after that give a picture about three different vulnerabilities that are common in the web services that we use every day. Exploiting these vulnerabilities is a crime and they are usually divided in to two different categorize. There are cyber assistant and cyber dependent crimes. Cyber assistant crimes are crimes that can be done without the assistance of it-equipment like fraud in the second-hand markets of the internet. The cyber dependent crimes are crimes that require it-equipment as a part of the crime in order for the crime to be possible. These are crimes like denial of service against an online store. The stakeholders that are a part of the cyber are the services that have vulnerabilities, the criminals that exploit the vulnerabilities and the victims that experience loses. The web services work by communicating with the user's browsers with hypertext transfer protocol messages and this is one of the key points in understanding how the web works. The criminals are divided in to six categories based on their abilities and motivations. The criminals are the unaware, casual criminals, hackers, professional criminals, hacktivists and nation states. The victims can be separated in to four groups based on their knowledge and how they become victims of a cyber-crime. The victim groups are gullible, greedy, inexperienced and unlucky people. This thesis focuses in three different web-based vulnerabilities that are injection vulnerabilities, broken authentication and cross-site-scripting. In injection attacks the attacker's goal is to inject code or database queries trough the user interface to be executed in the backend server. Broken authentication system vulnerabilities cause the authentication system to be vulnerable for attacks that make it possible to make changes to other user accounts. Cross-site-scripting vulnerabilities mean that the attacker is able to plant code in to the site that is then ran in another user's web browser. Nearly all of these vulnerabilities are caused by not handling user input properly in the servers properly.

Keywords: cyber-security, application security, cyber-crime, cyber assistant crime, cyber dependent crime, cross site scripting, SQL injections, broken authentication systems, web-based vulnerabilities

Janne Siltainsuu

The web-based vulnerabilities, stakeholders and avoidance

Vuosi 2018

Sivumäärä 29

Tämä opinnäytetyö keskittyy kyberturvallisuuteen ja web-pohjaisiin haavoittuvuuksiin sekä sidosryhmiin, jotka ovat osa näitä haavoittuvuuksia olemalla teknisiä edellytyksiä, hyökkääjiä tai hyökkäyksen uhreja. Tämän opinnäytetyön fokuksena on antaa lukijalle kuva ensin kyberavaruudesta ja sen sidosryhmistä ja sen jälkeen kolmesta yleisestä haavoittuvuudesta, jotka ovat yleisiä verkkopalveluissa, joita käytämme joka päivä. Näiden haavoittuvuuksien hyödyntäminen on yleensä aina rikos ja nämä rikokset jaetaan kahteen kategoriaan. Kyberavusteisiin ja kyberriippuvaisiin rikoksiin. Kyberavusteiset rikokset ovat rikoksia, jotka voidaan toteuttaa ilman tietokoneitakin, mutta ne mahdollistavat kommunikoinnin. Tällaisia rikoksia ovat esimerkiksi käytetyn tavaran verkkomarkkinapaikalla tehty petos. Kyberriippuvainen rikos on rikos, jonka toteuttaminen ei ole mahdollista ilman tietokoneita. Tällaisia rikoksia ovat esimerkiksi palvelunestot. Sidosryhmät, jotka ovat osa verkkopalvelua ovat verkossa toimivat tekniset laitteet, joissa on haavoittuvuuksia, rikolliset, jotka hyödyntävät näitä haavoittuvuuksia ja uhrin, jotka kärsivät haittaa rikoksien vuoksi. Verkkopalvelut toimivat kommunikoimalla hypertext tranfer protokollalla käyttäjän selaimen ja palvelimen välillä ja se on yksi tärkeimmistä tavoista ymmärtää miten tietoverkko toimii. Rikolliset jaetaan kyvykkyyksien ja motiivien perusteella kuuteen eri kategoriaan, jotka ovat, tietämättömät, välinpitämättömät, hakkerit, ammattirikolliset, haktivistit sekä valtiot. Rikoksien uhrin jaetaan neljään eri kategoriaan sen perusteella, miten he valikoituvat rikoksen uhriksi. Nämä neljä kategoriaa ovat, herkäuskoiset, ahneet, kokemattomat ja epäonniset. Tämä opinnäyte työ keskittyy kolmeen eri haavoittuvuuteen, jotka ovat erilaiset injektio haavoittuvuudet, rikkinäiset tunnistus järjestelmä haavoittuvuudet sekä cross-site-scripting. Injektiohyökkäyksillä tarkoitetaan hyökkäyksiä, jossa hyökkääjä onnistuu syöttämään käyttöliittymän läpi ohjelmakoodia tai tietokanta kyselyitä taustalla toimivalle palvelimelle, joka suorittaa nämä käskyt. Rikkinäiset tunnistautumisjärjestelmät, tarkoittavat haavoittuvuuksia, joiden avulla on mahdollista tehdä muutoksia muiden käyttäjien tileihin. Cross-site-scripting tarkoittaa mahdollisuutta syöttää sivustolle sisältö, joka suoritetaan uhrin selaimessa. Käytännössä kaikki haavoittuvuudet johtuvat huolimattomasta ohjelmoinnista ja syötteiden tarkistamatta jättämisestä.

Avainsanat: Kyberturvallisuus, sovellusturvallisuus, kyberrikos, kyberavusteinenrikos, kyberriippuvainenrikos, cross-site-scripting, SQL injektio, web-pohjainen haavoittuvuus.

Table of contents

1	Introduction.....	6
2	The stakeholders in web-based vulnerabilities.....	10
2.1	Web service.....	10
2.2	Attackers.....	11
2.3	Victims.....	13
3	Technical vulnerabilities.....	15
3.1	Injections types.....	15
3.1.1	Attack methods and goals.....	16
3.1.2	Exploitation example.....	17
3.1.3	Avoiding injection vulnerabilities.....	19
3.2	Broken authentication and Session Management.....	20
3.2.1	Description.....	20
3.2.2	Example of a broken authentication system.....	20
3.2.3	Avoiding the vulnerability.....	21
3.3	XSS - Cross site scripting.....	22
3.3.1	Attack methods and goals.....	22
3.3.2	Example of cross-site-scripting attack.....	23
3.3.3	Avoiding XSS vulnerability.....	24
4	Conclusion.....	25
	References.....	26
	Figures.....	28

1 Introduction

During the internet age we live in where internet and its services are a part of our everyday lives. These services have become a natural environment for all kinds of cultures and people to interact with each other. (Kritzinger & Von Solms, 2010) Services that earlier demanded us to be physically present now can be done in the internet without being depended on time or geographical location. The internet has been able to provide us an infinite source of information, the possibility to communicate with other people in few seconds to anywhere in the world and the first truly global market place where services and products change owners every second. It is said that nothing good can exist without the presence of bad and the internet is no exception in this. The cyber environment has also brought new types of risks in to our daily lives that we should consider. The criminals have always followed to the domains where people spend their time and money, so it is a natural path to follow people in to the internet, because that is where normal people consume their time and money. The wide spread of information systems on all kinds of businesses has also drawn the attention of criminals and other hostile actors. (Backhouse & Dhillon, 1996) It is essential that all the actors that spend their free time, work or develop internet systems have the understanding about the risks in the modern information systems. Only understating the risks makes it possible for these actors to understand how the consequence if they do not protect their information in the right way. (Kritzinger & Von Solms, 2010) As in all domains of crime also the cybercrime is always a combination of motive, opportunity and ability do the criminal act. In a way cyberspace can be seen as an interface for a traditional crime. (Felson & Clarke, 1998)

The cost of the cybercrime for the individuals, businesses and societies are generated through the damage that the cybercrime causes. This damage is the generated through affecting import and export, the decrease in sales, investments that need to be taken to increase security, the decrease in innovation because of security improvements. Cost of cybercrime is also a part of the larger effect on the world economy. The estimates for the monetary value varies and there is no real sum of money that can be verified, since the criminals do not pay taxes. However, in the year 2014 a study revealed that the cost of cybercrime is about 0,8% of the whole worlds gross domestic product. The significant of the cybercrime is easily seen when it is compared to the amount that the drug industry affects which is 0,9% of the global GDP. (McAfee, 2014) When understanding these crimes there is a difficulty since criminals to not pay taxes and most of the victims do not report these crimes. A study reveals that only one out of five victims reports the crime. (Symantec, 2011) This is due that individuals usually do not consider anything to be gained from reporting the crime, since the criminal is usually from another country or very often from a completely different continent. The companies usually do not report these crimes, since that might have a significant effect on their business, since the customers trust is damaged and even though the crime would be reported the same reasons as individuals make it appear as a situation where there is nothing to gained and the sad part is that, that is usually true.

This study that was done as a literature review examines the common ways of implementing cybercrime through using web-based vulnerabilities. The goal of this study is to reveal the basic ways to implement attacks through web-based vulnerabilities. The goal is that after reading this study the reader has a brief understanding about how these web-based vulnerabilities work and how can they be mitigated. We will focus on three different vulnerabilities that are quite common in the web. These vulnerabilities are injection vulnerabilities, broken authentication systems and cross-site scripting. The reason to choose these three are because the injection vulnerability attacks the logic of the service, cross-site-scripting attacks the end user and the broken authentication systems is in a way between the service and the end user, so they attack both.

The study will focus on finding answers to the following three questions:

- What are web-based vulnerabilities?
- Why do web based vulnerabilities exist?
- How can web-based vulnerabilities be avoided?

In this study when talked about information security, it means the confidentiality, integrity and availability of information and services. Confidentiality means that the information must remain secret to parties that do not have the authorization to view the information. The integrity of the information means that information must not be altered without proper authorization and the information is correct. The availability means that the information must be available to the parties that have authorization to view it when they need the information. This is also known as the standard information security triad and is widely used. (Peltomäki & Norppa, 2015)

During the study the term cyber security will be used often. This means a wider part of the information security and it means the state where information security can be trusted in general. It means a state where the general public trusts the systems that secure their information and they generally choose to use information system services. This state has been generally seen in Finland where people choose to use online banking and paying with credit cards and consider it to be more secure than using physical banks or having cash. (Peltomäki & Norppa, 2015)

During the study modern information society will be mentioned frequently. Modern information society means a society where people have access to the information networks, such as the internet. A society is considered to be a modern information society if the everyday services are provided in the internet. These services are provided to the customers as remote services and they are services like online shopping, banking, governmental services like taxing and social service. So, if considered in a simple manner, these are services in which the provider of the service and the customer never meet in person, but still they interact. The provider can also be information system and it does not require the other party to be

a living person. These services can be either services that the customer has to pay for or they can be without a charge. (viestintävirasto, 2015)

During this study critical infrastructure is considered to be the part of infrastructure that keeps the society running. It is considered to be energy services that create energy such as electricity and the powerlines that bring the power to the end user. Critical infrastructure is also the services that provide clean water and the services that purify water after it has been used. Logistic services such as road, harbors and airfields are also considered to be a part of critical infrastructure. The part that we are mainly interested in this study is the information systems and the cables and other parts that focus on delivering the information from the provider to the end user. (Valtioneuvoston kanslia, 2010) The main part of critical infrastructure is that most of them are provided with using information systems and nearly all of them require the work of information systems to provide critical infrastructure to the society. Without information systems it would be impossible for them to function properly or at all. (Hoffman, Rosenberg, & Washington, 2005)

During the study cyber-crime will be mentioned frequently. Cybercrime is generally an act that is forbidden in the law and it is done with the assistance of an information system. (Nykodym, Taylor, & Vilela, 2005) The motive through the cybercrime varies. Usually the motive is benefit monetarily from the crime either directly by gaining money or indirectly by gaining services that are not free without paying for them. Some cybercriminals have no intention of gaining monetary gains, but their motive is to gain reputation for themselves or to gain visibility for a political, ideological or religious view. Cybercrime can be specified in two different categories which are cyber dependent and cyber assistant crimes.

Cyber dependent crimes are crimes that require the computers and information systems as a part of the crime and they are not possible to be done without information systems. An example of a cyber dependent crime is a denial of service attack where an online shopping site is denied service during a busy shopping day like black Friday. This kind of a crime is not possible without using computers and information systems. (Viestintävirasto, 2016)

Cyber assistant crime is a crime that could be also done without the assistance of information systems. These are crimes like online fraud in the web's secondhand markets. A person can be tricked in these also in the real world and the technical side of the crime is not very sophisticated. In these kinds of crimes usually the information systems only connect the victim and the criminal. A very classical example of these are phishing emails, which could also be sent as physical letters asking for credit card details, so the email and information systems are only assisting the crime, but the crime does not require these systems to be successful. (Viestintävirasto, 2016)

During this study we will use the word vulnerability and weakness. Vulnerability is a flaw in the system that can be exploited and through that the system works in an unintended way. A weakness is a flaw in the system that cannot be exploited directly but it benefits the attack in one way or another to attack the system further.

In this study we will first focus in understanding how the applications, work and how the crimes are done through these services in a general manner. After this we will focus a little in to understating the victims and the attackers. After this we will focus on the technical side of these crimes and how they could be avoided.

2 The stakeholders in web-based vulnerabilities

When considering about web-based vulnerabilities there are three stakeholders. There needs to be a web service. The web service usually is a combination of different servers that all have a specific job. They include the operating system of the server, the webserver software, database servers and different backend processing units that do the calculations for the services. The web-based vulnerabilities always need an attacker that exploits the vulnerability that causes unintended behavior in the service. The motivation of these attackers varies from protesting and gaining services for free into gaining monetary value from selling and stealing databases or physical products that are gained by exploiting vulnerabilities. Although cyber-crime usually is seen as a victimless crime there always is a victim. Victims usually become victims because they are unlucky but there are also other situations where they become victims.

2.1 Web service

The web services work by communicating through hypertext transfer protocol (HTTP) messages. These are small messages that are delivered from the user to the web server and from the webserver to the user. These messages can also be encrypted and then the communication is secured (HTTPS). These communication methods communicate to a different port in the server. HTTP communicates to the standard port 80 and the HTTPS communicates to the standard port 443. When the users type in to the address bar of the web-browser "http://google.com" he sends the following message with HTTP protocol to the googles server:

```
GET / HTTP/1.1
Host: google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

In a simplified way the web browser says to the google.com server, GET me google.com. Then the http request travels through the internet and it is received by the google.com web server. In the server there are many operations that are performed during this operation. The web server software starts to build the website for the users based on the request. This request is not very complicated so then the users is built the standard google.com web page. Then web server then responses to the request by sending back the web-site to the user's browser with following response from the server:

```
HTTP/1.1 200 OK
Date: Mon, 23 Jul 2018 16:06:14 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=86400
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-07-23-16; expires=Wed, 22-Aug-2018 16:06:14 GMT; path=/; domain=.google.com
Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
Connection: close
Content-Length: 215293

<!doctype html><html > (after this the website is displayed in HTML form)
```

This is the basic way the web services work. The users send them requests in hypertext transfer protocol and the web server responses that back to them. In many cases the user logs in to the service and then the same kind of a request is sent, but this time the web server looks if the user's data matches the data in the database. If they do, the user is then logged in to the service and if not, then the user is returned with an error. (MacBeth, 2004)

2.2 Attackers

For a vulnerability to become involved in an attack there always needs to be an attacker to find and exploit that vulnerability. Attackers that are usually referred as cyber criminals work for motivation to gain something. Usually criminals are divided into six different categories on their knowledge and motivations. These six different groups are unaware, casuals, hackers, professional criminals, hacktivists and nation states. (Peltomäki & Norppa, 2015) These kinds of groups are usually also found from other sources, the famous Finnish cyber security speaker Mikko Hyppönen (2012) also uses same kind of a grouping with the difference that he sees all the criminals under the name criminals, but hacktivists and nation states have motives that differ from them so they are still their own category. (Hyppönen, 2012)

The unaware criminal usually is a person who either does not know or understand that the action is a crime, or they are working under another party and they believe that the act is legitimate. These people might perform a denial of service type of attack for a web service by thinking that it is a joke. They might also be just a proxy for the actual criminal to perform the exploitation of the webservice. Usually this type of attacker is not so

involved in breaching a web service, they usually work as mules that launder money or other goods for the criminals in the post phase of the crime. (McAfee, 2014)

The casual criminals are criminals who do not consider their action to be a crime. (Peltomäki & Norppa, 2015) Usually these kinds of criminals are quite young, and their main motivation is to gain some services or content for free that would otherwise require paying for it. An example of this kind of a web-based attack would be a vulnerability that would let the attacker use a known easy exploit to alter the database authorization table to give access to a certain movie for free that would require paying.

Hackers are technically sophisticated individuals who usually test their skills in order to find vulnerabilities in services and software to break their security measures. (Peltomäki & Norppa, 2015) It is quite common to have the association to criminals when talked about hackers, although there are a number of people working with the title white hat hacker to make systems and services more secure. (Kovacich, 1999) The attacks that hackers can perform can vary from all the basic vulnerabilities to also social engineering which means hacking the human element of security in order to gain access to the system without actually touching the keyboard. This can be done with persuading an IT-staff member to perform the malicious action by telling them lies. (Mitnick, Simon, & Wozniak, 2011)

The group that causes most likely the biggest costs for organizations and individuals are the professional criminals. The professional criminals perform cyber-crimes always with the intent of making money with the crimes. These crimes usually are in one way or another tied to products and services that exist in the real world because they can be transferred into money easily. (Wueest, 2016) The interesting part about organized cyber-crime is that they usually are based on different subcontracting networks where hackers sell their hacking services, malware writers sell malware and so on, which then the professional criminals then buy in order to perform other crimes. (Jahankhani & Al-Nemrat, 2016) This kind of subcontracting network is almost impossible to start taking down since they usually do not even meet each other during the sales and they might be in totally different continents. An example of a web-based vulnerability that organized cyber-crime might exploit is a flaw that makes it possible for them to gain access to users' accounts on online stores and order products that then get shipped overseas.

Hacktivists are criminals who are not driven by monetary gain. The motivation for them is to perform non-violent activism in the internet in order to raise attention for ideological, political or religious views that they drive. These protests usually are either illegal or the work in the grey area of the law. (Peltomäki & Norppa, 2015) The interesting part about hacktivism is that they usually have no centralized leadership, and anybody can just start wearing their "uniforms" and start protesting whatever they want to. Also for this reason it is impossible to stop these organizations since there is no real organization. (Taylor, Jordan, & Samuel, 2004) Hacktivists are known for their denial of service campaigns and also by exposing different confidential materials in the internet. One of the examples of crimes that

hacktivists have traditionally done is to deface websites through technical cross-site-scripting vulnerabilities that take over the visible part of the site. These attacks do not usually affect the real information on the site, but they do make the users experience to appear as if the whole site has been hacked and by that way their intent is to crash the whole trust of the customer to the particular site. (Hampson, 2012)

The last subgroup of cyber criminals are the nation states. Nation states are in a way very special group of cyber criminals, since they are not criminals in their own jurisdiction. They always work in order to promote their own national interest. Nation states use their resources for intelligence, cyber sabotage and cyber vandalism. (Nguyen, 2015) Nation states usually do not mainly attack web-based vulnerabilities, since they only give you access to the information that the service handles. Usually nation states attack the backend server. In order to gain access to the backend server they might although perform some sort of a code injection attack to gain initial access to the back-end server.

2.3 Victims

In understanding the vulnerabilities there always needs to be a criminal that exploits the vulnerability and a victim that is affected in the crime. Every day about a million users become victims of a cyber-crime (Symantec, 2011) and cyber-crime affects individuals, organizations and businesses. (Nykodym et al., 2005) According to Joseph (2006) most common victim types are grouped in to four different groups. These groups are gullible, greedy, inexperienced and unlucky people.

The gullible people are people who sincerely believe that they have won something or that they have received an offer that is too good to be true. Most commonly gullible people are either very young or very old and they might not have the necessary understanding about what are the risks when using the networks. (Joseph, 2006) When looked about web-based vulnerabilities gullible might become victims of a cross-site-scripting attack that steals their credentials by tricking them in to giving them to the attacker. Usually the gullible people are home users since organizations usually train their staff in to detecting these kinds of attacks and scams. (Kritzinger & Von Solms, 2010)

The greedy victims usually are drawn by fast and easy monetary gains. These victims are usually do know what kinds of scams are in the networks, but they are blinded by the gains that they think they are gaining. These people usually can be told by others that they are getting scammed, by still they believe that the gain that they are about have is sincerely true. In extreme cases greedy victims might even take loans or sell their own belongings to pay the scammer. (Joseph, 2006) Greedy victims usually become victims of cross-site-scripting attacks that scam in some way. Since this is a psychological attack vector the technical know-how usually does not play a part in getting scammed. (Kritzinger & Von Solms, 2010)

The inexperienced victims are usually victims that might not even notice that they have become victims of a cyber-crime since their technical know-how and understanding about technology is very limited. (Joseph, 2006) Inexperienced people are usually young or very old. They usually also do not have any training from an organizations. (Kritzinger & Von Solms, 2010) The inexperienced victims can become usually victims of all kinds of vulnerabilities and malwares since they do not have any means to protect themselves.

The unlucky people are usually the people who just happen to be at the wrong place at the wrong time. Usually when they become a victim there was nothing they could have done other vice to avoid becoming a victim of a crime. Basically everybody can be unlucky victim since they usually are a part of a larger security breach. (Joseph, 2006) Usually they become victims of security breaches that could be done through an injection attack that enables for the whole database of a service to be stolen.

3 Technical vulnerabilities

In this part we will focus in three different types of vulnerabilities and the technical implementation of these vulnerabilities. These three different vulnerabilities are injections, broken authentication systems and cross-site-scripting vulnerabilities. Injection vulnerabilities mean that it is possible to inject database queries, programming code or other kind of information through the user interface that is later executed in the backend server to cause an unintended behavior. Broken authentication systems mean that there is some kind of a vulnerability in the session management or user authentication system. Almost all services require the user to login to view their personal content on the service and one of the most serious events of a system is if another user is able to view other user's information or alter it. These are usually done through some kind of a flaw in the authentication system. The third vulnerability is cross-site-scripting also known as XSS. Cross-site-scripting vulnerabilities work in a way that the attacker is able to plant JavaScript code in to the site through a form and then that script is executed in the victim's web browser. So, this means that the attackers were able to do "cross-site-scripting" to the victims browser. The three vulnerabilities were chosen because the injection vulnerabilities attack the backend server or service and the cross-site-scripting attacks the end user through the frontend of the service. The broken authentication is a kind of a hybrid since it attacks the user and the backend service depending on how the attack is done.

3.1 Injections types

Injection vulnerabilities include SQL (Structured query language) injections, code injection, operating system injections and all injections that enable the target system to perform an operation that it is not intended to perform. In injection vulnerabilities the attacker can trick the target system to modify, delete or leak out information without proper authorization. (OWASP, 2013) These attacks are performed because of various reasons, but common to all of the attacks are that the attacker usually has some intention in executing these attacks. The attacks usually start with a reconnaissance phase in which the attacker gathers information about the service and tries to find possible injectable parameters. These parameters are usually found from different parts of the application where the user is able to provide input to the application. (Halfond, Viegas, & Orso, 2006)

SQL (Structured query language) injections refers to a situation where the attacker is able to inject database commands to the backend server through the application in the frontend that is handling the service. This is usually done through a html form that processes user input. (Shakya & Gupta, 2016)

Code injection refers to a situation where the attacker is able to inject code in to the application or server and then the code is interpreted or executed by the application. (Open Web Application Security Project, 2013)

Operating system injections refer to a situation where the potential attacker is able to inject operating level commands through the frontend application to the backend server where the commands are executed.

3.1.1 Attack methods and goals

Attacks can be produced in many different ways and with many different goals. Usually the goal is to gather some information that has value to the attacker. This information can be user data like usernames and passwords, social security numbers, credit card or banking information. All of this is data that the service has to have at some form in their database to produce the service that they are providing. This is also the reason why services that sell products, provide healthcare or other services that have information mentioned before are certain to be attacked at one point or another.

One of the basic ways to attack the server with injection attack is the fingerprint attack. The attack means that by injecting database query, code or operating system command the server gives out information about the database, service or operating system. One of the goals can be to try and get the service to crash and cause an error message. This messages quite often contain some information about the service. This is known as a "fingerprinting attack" and it can be done to the database in SQL injection, to web service running and to the operating system. (Halfond et al., 2006)

Attacks might not be interested in the data that the service contains and because of that they might just be interested in denying service for all the other users. This kind of situation might cause huge losses to a company in certain events that occur only on rare occasions. This kind of a situation of denial of service could be the betting on super bowl or the black Friday sales before Christmas. If the service is down during these times it might cause huge losses to the company that is being attacked. This kind of application denial of service attack through an injection could be caused by injecting code to the server that causes the whole system to crash every time it is restarted. It is also possible to crash the whole server by removing all the data in the database. In this case it would be done with a SQL injection and by causing the tables on the database to be dropped. This might cause huge losses because many services are dependent on certain information that is stored in the databases. (Halfond et al., 2006)

Looking into SQL injections they usually have four goals and they all are basic database interactions. SQL injections look in to creating new data, reading old data, modifying old data and deleting data from to server. These are all basic database interactions and they all have a harmful way to be used if the attacker is able to by-pass the security measures in the service. In the case of creating new data, the attacker might be interested in adding a new user to the service that has all the privileges to interact as an admin. The attacker might be interested in reading all the data stored in the database in order to steal user related data. The attacker might be interested modifying application data to have access to certain

information or to get certain parts of the service without the need to pay for the service. The attacker might be interested to delete certain data from the service in order to cause some behavior or to denial service from other users. (Halfond et al., 2006)

The attacker could also be interested in performing code execution on the server in order to access or modify certain functions in the service. These actions might be to have access to certain information, gain foothold to the server to be used in later attacks or privilege escalation in the service or server. (Halfond et al., 2006)

3.1.2 Exploitation example

The very basic example of a SQL injection is to inject " ' or 1=1; -- " to a html form that gets passed to a database server. This means that we are going to first interrupt the statement with the character " ' " and then pass our own command to the database server which in this case would be " or 1=1; --". This means that since there usually is some sort of a true or false statement the 1=1 will always be true and this would possible allow us to perform our intended malicious action. In the case that the SQL command that we would use to find a certain user that is called by the name that we deliver in parameter \$username would be the following:

```
SELECT * FROM users WHERE username LIKE '$username';
```

The intended situation would be that we would deliver a name, like "bob" to the query and the query to the database would look like the following:

```
SELECT * FROM users WHERE username LIKE 'bob';
```

But in this malicious intent of injecting the string " ' or 1=1; -- " the query would look like the following:

```
SELECT * FROM users WHERE username LIKE ' ' or 1=1; -- ';
```

So in this example we would look for a username that is empty or a condition that is always true and everything after the two lines will be discarded by the database engine since it is considered to be a comment. This could lead in to the situation that the database engine would deliver all of the usernames to the attacker.

In the real example we are going to be doing and SQL injections to a website that is hosted on our own virtual network. The virtual testing environment is OWASP's web application testing environment and it is intentionally vulnerable to SQL injections.

The form that we are going to be injecting our payload is a search html form that is intended to be used to look up players in the system. The form then passes the requested name to the backend server that handles the request. The request is not validated and the request is not parameterized. This means that anything that the potential attacker

writes to the HTML form is then passed to the backend database server. In this example the form takes in the user input which then is passed to the variable "players" and after that by using post request transmitted to the server to be parsed and executed:

HTML form that we are using is the following:



The image shows a web form with a light blue header containing the text "You can search for your favorite Guessnum player by entering the play". Below the header is a form with a label "Guessnum Player:" followed by a text input field and a button labeled "SEARCH".

Figure 1 - Example player search form

The form in HTML markup:

```
form action="guessnum5.php" method="post">
    Guessnum Player: <input type="text" name="player" size="30">
    <input type="submit" value="SEARCH">
</form>
```

Once this form is submitted the post request that is going to be sent to the server is the following:

```
POST /vicnum/guessnum5.php HTTP/1.1
Host: 192.168.12.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.12.10/vicnum/guessnum.html
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 25

player=*
```

We are going to pass this request to a tool which is called sqlmap and it is built to be used as a tool to work with servers that are vulnerable to sql injections and to steal data from these servers.

In this example we are going to save the request to a file called request.txt and then run the following command to the sqlmap software on the linux command line:

```
sqlmap -r request.txt --passwords
```

This command enumerates the database by using the sql injection vulnerability in the form after that it finds the passwords in the databases and then it uses a basic dictionary attack to crack them in a cleartext format:

```
database management system users password hashes:
[*] bricks [1]:
    password hash: *255195939290DC6D228944BCC682D2427DA57E21
    clear-text password: bricks
[*] bwapp [1]:
    password hash: *63C3CE60C4AC4F87F321E54F290A4867684A96C4
    clear-text password: bwapp
[*] citizens [1]:
    password hash: *E0E85D302E82538A1FDA46B453F687F3964A99B4
[*] cryptomg [1]:
    password hash: *2132873552FEDF6780E8060F927DD5101759C4DE
    clear-text password: cryptomg
```

Figure 2 - SQL injection result using sqlmap

3.1.3 Avoiding injection vulnerabilities

Avoiding injection vulnerabilities is done by three basic steps which usually work to nearly all web-based vulnerabilities. The steps are input validation and in database queries parameterized database queries.

Validating input means that the input that users send to the web service is validated that it does not contain any harmful and the information is what the application expects it to be. For an example if the user is supposed to send a phone number to the site it will only contain numbers and in extreme cases it might have a country code that has a plus in front of it. This means that there is no reason to send anything else than numbers with or without a plus character. If the backend server checks in this case that is the user sending in only numbers, then there is not a possibility that the user could inject code or database queries to the system. In other cases, the input could be validated in a way that special characters in attacks are not allowed.

Best practices to prevent database injections is to use parameterized database queries. This is also the way every developer should be writing their database queries from the beginning. The basic way is to first define the query beforehand and after that every parameter is passed to the function separately, this means that the query is parameterized. This function is usually built in to the programming languages. The only down side to parametrized queries is that they might in some extreme cases hurt performance. Although parametrized queries tend to be safe, they should always be used with validating.

3.2 Broken authentication and Session Management

Every application that the user interacts with needs a way to identify the user and provide only the certain type of content to the user. The most common way to authenticate a user is to provide password and username pair. They are matched to a password and username located in the services database and if they match the user is given a unique token, usually a HTTP cookie that the user is recognized by the service. (Stuttard & Pinto, 2011)

3.2.1 Description

Broken authentication means that the authentication can be tricked to allow users that are not authenticated to interact with the service. The session management means how the service handles session information. These can often be exploited in ways that allow the attacker to compromise passwords, keys or session tokens. Flaws in authentication and session management also may give the attacker the ability to take other users identities. (OWASP, 2013) Authentication and session management creates the foundation to nearly every service that users use every day. The session management has to be implemented in a way that the session id is not easily guessable, or it is not a hash of a known word like the user name. The user's session token also needs to be verified during every state altering request.

3.2.2 Example of a broken authentication system

As mentioned before the checking of the who the user is based on different factors, usually a cookie that is given by the web server and the user agent sends it as a part of every request to the server and that way the web server knows who it is talking to. This is a very traditional way of setting the cookie in a response that is coming from a server:

```
HTTP/1.1 200 OK
Date: Tue, 24 Jul 2018 17:40:19 GMT
Set-Cookie: PHPSESSID=vimhk6iasaggxfdsvl24772; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
```

The example of using a broken authentication system and session management would be if the user's cookie would be set always the same because it would be for an example the users id number in the database. This is not very commonly seen anymore, but some applications might have some parts that recognize users by setting their cookie to be their unique id in the database and hashing that with a weak hashing like md5. So if the users id number would be 111222 the generating of the hash would be "00b7691d86d96aebd21dd9e138f90840" and if this would have been discovered by a hacker they could just set their own cookie with any users cookie by discovering their id number and hashing that with md5.

3.2.3 Avoiding the vulnerability

Session tokens should not be easily guessable, or the session token should not be a weak hash that can be reversed easily. The session token also should not be handled insecurely. An example of handling session tokens insecurely is handling them in http-get requests, which means that the session token is transported and for that reason visible in the address. (Owasp, 2017)

The service should detect and prevent brute forcing attacks on the system. Brute force attack means an attack where the attacker goes through a premade list of usernames and passwords trying to find the right one. This can be prevented by using a simple fail to ban function that prevents login for a short time period if there are too many failed tries. (Owasp, 2017)

The service should also rotate the session token and unvalidated it on the server side on logoff. This needs to be done in order that the attacker cannot try to log back-in using the users old session token. (Owasp, 2017)

3.3 XSS - Cross site scripting

Cross site scripting means the ability to run scripts on other users' browsers by injecting certain kind of code in to the site. This gives the attacker the possibility to steal information or make other users perform certain types of actions on a service that the user did not intend to do. (Stuttard & Pinto, 2011) With cross site scripting the attackers are able to steal other users sessions, deface websites to cause humiliations or redirect users to sites which contain malware or disturbing content. (OWASP, 2013)

3.3.1 Attack methods and goals

There are three types of cross site scripting types. These are Reflected, Stored and Document object model based. There are differences in the ways how these attacks work and how they can be implemented, but the aim is the same. Injecting malicious content through the site to the victims browser and get it to be executed in the victims internet browser.

Reflected Cross Site Scripting means that the web application or application programming interface will accept unvalidated or unescaped input as part of the websites document form. When the attacker is able to inject this to the website and then this script is executed in the user's web-browser. Usually reflected cross site scripting requires the user to interact with the system in some way. This way can be viewing of a malicious advert is meant or clicking a malicious link that the attacker has posted to the site. (OWASP, 2017)

Stored Cross Site Scripting means that the application or application programming interface stores the malicious JavaScript in some way. Usually this means that it is stored in a database. When this happens, the malicious JavaScript is usually displayed and also executed in a victim's internet browser when the victim visits the website. The stored cross site scripting is considered the most dangerous since it can have a massive effect to a large number of users. (OWASP, 2017) An example would be the possibility to inject malicious JavaScript to a comment field of kim kardashians instagram account. This would lead in to attacking hundreds of thousands of users in a very short period of time.

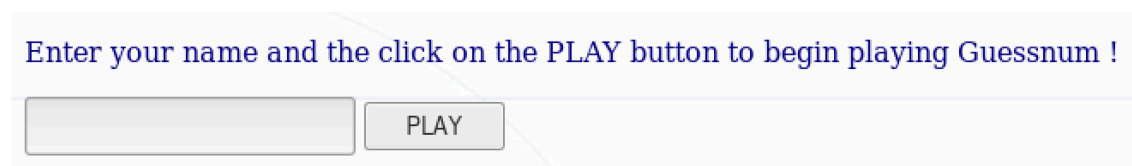
Document object model based cross site scripting means that the web application dynamically includes attacker-controlled content to the page. This means that the attacker is able to inject malicious JavaScript to the application and that attack against the users. (OWASP, 2017)

Usually this cross site scripting attacks try to attack users or the administrators. The attacks include attacks like stealing session cookies, stealing user accounts, multi factor authentication bypassing and injecting malicious html content to the pages, which usually try to download a malware to the victims computer. (OWASP, 2017)

3.3.2 Example of cross-site-scripting attack

In this example that is done by using a virtual environment and the OWASP test environment we are going to illustrate how the stored cross site scripting works. In this example we have a login form that takes in user submitted information but does not validate the input in any ways. The form that we will be examining in this example is the following and we are interested in the following variable **player**.

The form in the web-browser:



The form in html markup:

```
<form NAME="g" ACTION="cgi-bin/guessnum1.pl" METHOD="post">
  <input type="text" name="player" >
  <input type="hidden" name="admin" VALUE="N">
  <input TYPE="SUBMIT" VALUE="PLAY">
</form>
```

The intended behavior is that, the user submits their name and the name is then the name is posted on the next page. This does happen, but the situation is that the backend server does not validate the input coming in inside the post request. So when the potential attacker injects the following payload "`<script>alert("XSS");</script>`" to the form presented in the picture, it travels first to the backend server and then it gets printed to the next page as a html element that looks like the following:

```
<h2>Welcome <script>alert("XSS");</script> - the computer has chosen a number </h2>
```

The result that the victim will see in this particular case would look like this:

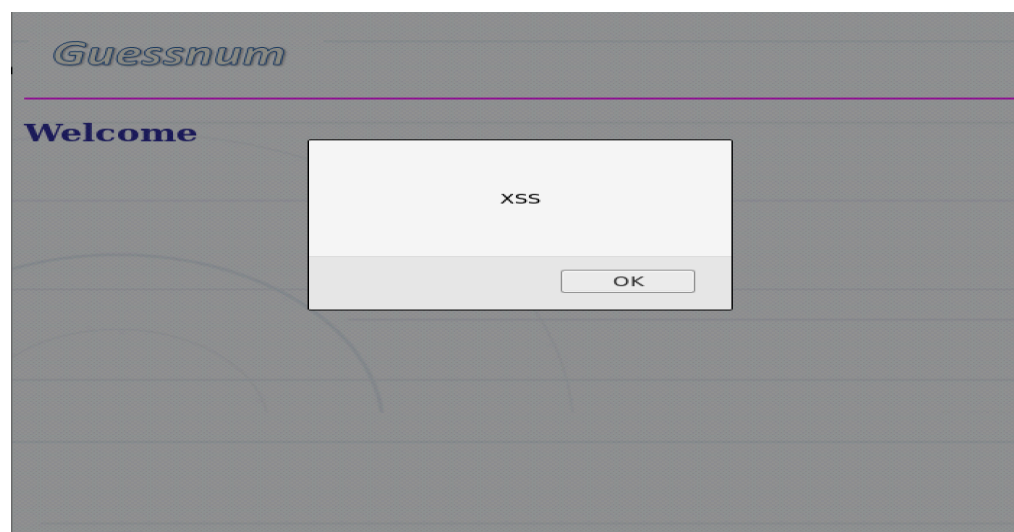


Figure 3 - XSS payload executed on the browser

In this example we have successfully triggered a cross site scripting vulnerability.

3.3.3 Avoiding XSS vulnerability

The cross-site scripting vulnerability is real nearly all the times because the application does not properly validate and escape user input to the site. This means that the application should always check the input that the user gives to the site and determine does it hold malicious characters. The vulnerability can also be mitigated with using different web frameworks that are up to date. These frameworks include software like, ruby on rails, react, JavaScript and WordPress which are designed to escape malicious input by design. The application should also escape untrusted data in the http requests. This untrusted data contains different body attributes, JavaScript, cascading stylesheet information and unified resource locator data. This escaping will resolve most of stored and reflected cross site scripting vulnerabilities. (OWASP, 2017) Also adding the content security policy header will add another layer to protect against cross site scripting attacks. Content security policy is another layer to be added in order to help detect and mitigate attacks. It also helps to mitigate other attacks than cross site scripting. The content security policy specifies what are the domains that the users browser can consider as trusted sources of valid executable scripts. (Medley, 2018)

4 Conclusion

This thesis has focused in understanding cyber-crime and how the web-based vulnerabilities are a part of them. The web-based applications work by getting requests from the users that is then transmitted usually as a HTTP request to the server and then it is processed there to cause some behavior in the application. A traditional example would be a login form that asks for the user to give a username and a password that are then transmitted in an encrypted https request to the backend server where it is then compared to the user details in the database, if they match the user is then sent back the web site as a logged in user and if not then the user receives an error message back.

The attackers in the world represent six different groups that are separated by their motives and technical knowledge. These six groups are unaware criminals, casuals' criminals, hackers, professional criminals, hacktivists and nation states. Unaware perform crimes without even understanding that their actions are considered to be crimes, or they are a being scammed by a third party. The casual criminals do not consider their actions in being crimes because they are not very serious crimes. The hackers are technically sophisticated persons who use their knowledge in breaking in to services. The professional criminals seek monetary gains from their crimes and they try to run it as a business. Nation states seek their national interests and their main goal is to find intelligence information about other states. (Peltomäki & Norppa, 2015)

The three main vulnerabilities that were discussed were injection vulnerabilities, broken authentication systems and cross-site-scripting. Injection vulnerabilities mean that through the user interface the potential attacker is able to inject programming code, database queries or other information to the backend server that gets executed there and causes unintended behavior. Broken authentication systems are vulnerabilities that lets the potential attacker to view another user's information and possibly alter it. They usually exist because the backend server does not check the session information, or the user identification system is poorly designed. Cross-site-scripting means that the potential attacker is able to input JavaScript code in to the service that then gets executed in the other victim's browser. This might cause the user to be redirected in to a malicious site in order to install malware or trick the user into giving their user information.

The cause of these vulnerabilities and all the other vulnerabilities are programming and system design errors that have been made and not found by the organization that is offering the service for the users. All of the vulnerabilities are caused by not checking the users input in the backend servers. The means of checking that the input is valid is to limit the possibility of characters inputted. If the service asks for the persons age then only numbers between zero and 120 are possible inputs, there is no reason to accept information that might be letters.

The world is under a constant change and the web-based vulnerabilities change constantly. Future research could focus in vulnerabilities in the modern web-applications.

References

- Backhouse, J., & Dhillon, G. (1996). Structures of responsibility and security of information systems. *European Journal of Information System*, 5, 2-9.
- Felson, M., & Clarke, R. V. (1998). Opportunity Makes the Thief Practical theory for crime prevention.
- Halfond, W. G. J., Viegas, J., & Orso, A. (2006). A Classification of SQL Injection Attacks and Countermeasures. *Proceedings of the IEEE International Symposium on Secure Software Engineering*, 1, 13-15. Retrieved from <http://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf>
- Hampson, N. C. N. (2012). *Hacktivism: A New Breed of Protest in a Networked World*. *Boston College International and Comparative Law Review* (Vol. 35). Retrieved June 20, 2018 <http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authType=crawler&jrnl=02775778&AN=80131321&h=Gsl8H9LfdhyApuxPsS1uhzJxMrqRrUJQy7BbRCTFklc%2F0kem8cGzRRS1qaXAhm27f1VlO5Wz2LqvqZZ6DRB7Hg%3D%3D&crl=c>
- Hoffman, L. J., Rosenberg, T., & Washington, G. (2005). Types of cyberexercises. *IEEE Security & Privacy* 3.5, 27-33.
- Hyppönen, M. (2012). The three types of online attackers - TechRepublic. Retrieved February 19, 2018, from <http://www.techrepublic.com/blog/it-security/the-three-types-of-online-attackers/>
- Jahankhani, H., & Al-Nemrat, A. (2016). Examination of Cyber-criminal Behaviour, (January 2012).
- Joseph, A. E. . (2006). Cybercrime definition. Retrieved March 12, 2018, from <http://www.crime-research.org/articles/joseph06/>
- Kovacich, G. L. (1999). Hackers: Freedom Fighters of the 21st Century. *Computers & Security*, 18, 573-576.
- Kritzinger, E., & Von Solms, S. H. (2010). Cyber security for home users: A new way of protection through awareness enforcement. *Computers and Security*, 29(8), 840-847. <https://doi.org/10.1016/j.cose.2010.08.001>
- MacBeth, G. S. (2004). Web Services. *C# Programmer's Handbook*, (Chapter 1), 397-405. https://doi.org/10.1007/978-1-4302-0797-9_18
- McAfee. (2014). Economic impact of cybercrime II.
- Medley, J. (2018). Content Security Policy (CSP) Title. Retrieved April 20, 2018, <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>
- Mitnick, K., Simon, W. L., & Wozniak, S. (2011). *Ghost in the Wires: My Adventures as the World's Most Wanted Hacker*. Little, Brown. Retrieved from <https://books.google.fi/books?id=p-nRxITKc34C>
- Nguyen, D. (2015). State Sponsored Cyber Hacking and Espionage.
- Nykodym, N., Taylor, R., & Vilela, J. (2005). PROFILING OF CYBER CRIME Criminal profiling and insider cyber crime. <https://doi.org/10.1016/j.clsr.2005.07.001>
- Open Web Application Security Project. (2013). Code Injection. Retrieved June 12, 2018, from https://www.owasp.org/index.php/Code_Injection
- Owasp. (2017). Top 10-2017 A2-Broken Authentication. Retrieved April 12, 2018, from https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication
- OWASP. (2013). Top 10 2013-Top 10. Retrieved October 17, 2017, from https://www.owasp.org/index.php/Top_10_2013-Top_10
- OWASP. (2017). Top 10-2017 A7-Cross-Site Scripting (XSS). Retrieved October 17, 2017, from [https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_(XSS))
- Peltomäki, J., & Norppa, K. (2015). *Rikos meni verkkoon : näkökulmia kyberrikollisuuteen ja verkkoturvallisuuteen*. Helsinki: Talentum. Retrieved from <https://jyu.finna.fi/Record/jykdok.1458178>
- Shakya, S., & Gupta, A. (2016). Concerns on information system and security audit, 2(Journal of Advanced College of Engineering and Management), 127-135.
- Stuttard, D., & Pinto, M. (2011). *The web application Hackers handbook 2*.
- Symantec. (2011). CYBERCRIME REPORT 2011. *World*, 1(650), 94043.
- Taylor, P., Jordan, T., & Samuel, A. (2004). *Hacktivism and Cyberwars: Rebels with a cause?* *Journal of Chemical Information and Modeling* (Vol. 53). <https://doi.org/10.4324/9780203490037>

- Valtioneuvoston kanslia. (2010). *Yhteiskunnan turvallisuusstrategia*. Retrieved February 17, 2018, from <http://www.yhteiskunnanturvallisuus.fi/fi/materiaalit>
- viestintävirasto. (2015). Viestintävirasto - Palveluista säädetään tietoyhteiskuntakaassa. Retrieved January 22, 2018, from <https://www.viestintavirasto.fi/kyberturvallisuus/palveluidenturvallinenkaytto/palveluntarjoajanyhteystiedot.html>
- Viestintävirasto. (2016). Viestintävirasto -Tietoverkkorikollisuus - rikoksia verkossa tai verkon avulla. Retrieved October 23, 2017, from <https://www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2015/05/ttn201506031327.html>
- Wueest, C. (2016). Financial threats 2015. *Security Response*, 1.0(March 22), 30.

Figures

Figure 1 - Example player search form	18
Figure 2 - SQL injection result using sqlmap.....	19
Figure 3 - XSS payload executed on the browser.....	23