Security Research Paper

Mureseanu Alexandru Gabriel  FHICT Semester 3

Contents

[Introduction 2](#_Toc62166591)

[Methods 3](#_Toc62166592)

[Questions 3](#_Toc62166593)

[Results 4](#_Toc62166594)

[Injection 4](#_Toc62166595)

[Security risks of injection 4](#_Toc62166596)

[How can your website be attacked using injection? 4](#_Toc62166597)

[What are the best practices to protect your web application from injection? 5](#_Toc62166598)

[What defines an injection secure application? 5](#_Toc62166599)

[Does injection affect my project? 5](#_Toc62166600)

[Broken Authentication 5](#_Toc62166601)

[Security Risks of broken authentication 5](#_Toc62166602)

[How can your website be attacked using broken authentication? 5](#_Toc62166603)

[What are the best practices to protect your web application from having a broken authentication? 5](#_Toc62166604)

[What defines an authentication secure application? 6](#_Toc62166605)

[How I overcame authentication flaws in my project 6](#_Toc62166606)

[Sensitive data exposure 6](#_Toc62166607)

[Security risk of data exposure 6](#_Toc62166608)

[How can your app be attacked using data exposure? 6](#_Toc62166609)

[What are the best practices to avoid data exposure? 7](#_Toc62166610)

[What defines an application with secure data? 7](#_Toc62166611)

[How I handle data in my project 7](#_Toc62166612)

[Cross-Site Scripting (XSS) 7](#_Toc62166613)

[Security risk of XSS 7](#_Toc62166614)

[How can your app be attacked using XSS? 7](#_Toc62166615)

[What are the best practices to avoid XSS? / What defines an application that’s XSS secure? 8](#_Toc62166616)

[Does XSS affect my project? 8](#_Toc62166617)

[How to prevent other security risks 8](#_Toc62166618)

[Security risks of a distributed system 8](#_Toc62166619)

[Snooping 8](#_Toc62166620)

[Denial of Service / Distributed Denial of Service (DoS/DDoS) 8](#_Toc62166621)

[Spoofing 9](#_Toc62166622)

[General good practices 9](#_Toc62166623)

[General bad practices 10](#_Toc62166624)

# Introduction

In this paper I will showcase the results of the research that I have conducted on Application Security in detail.  
  
With the online landscape evolving every day, there are more and more ways to expand and use the medium, sometimes the services such as online banking, online payment, etc., require important data that needs to be kept secret. This does not apply only to the mentioned services, as a password leak from a website can lead to more of your accounts being stolen.  
  
While cyber security is still new to me, I have had some experience in my previous semesters. In the first semester we had to make sure that the connection between two applications is secure and in the second semester we had to make sure that user data was being encrypted.

In this semester there was a higher priority on consumer ethics so good security was a must in the projects that we made. This meant that I had to do research on the topic and find the best way to handle it, which is what I will showcase in this paper.

# Methods

The methods that I used to gather the data and information for this paper are the following:

* Self-research and application of the research based on the semester requirements.
* Scientific articles on the subject provided by experts in the scene.
* Implementation and testing of the methods found.

# Questions

Due to the huge quantity of data gathered during the research, the organization of questions will have to be a bit different

At the start of my research, I set out to answer the following question:

1. What are the most common security flaws in web applications?

Also, for each type of flaw the following questions needed to be answered.

* What are the security risks for a web app?
* How can your website be attacked?
* What are the best practices to protect your app?
* What defines a secure application (regarding the flaw)?
* Does this affect my project? / How I overcame this flaw.

1. What are the security risks of a distributed system?
2. What are some general good practices for safety?
3. What are some general bad practices for safety?

# Results

## Injection

### Security risks of injection

Injection is to this day the most widespread flaw in web development, these flaws are most found in SQL, LDAP, XPath or NoSQL queries, XML parsers and ORM queries.  
  
The impact of this flaw can result in data loss, corruption, or denial takeover. In extreme cases it can lead to complete host takeover

### How can your website be attacked using injection?

Most hackers will abuse vulnerable queries in your database communication where the query is not sanitized   
Assume we are passing a parameter from the front end to communicate with the database;  
if the parameter passed down to the query is not sanitized the hacker can run code in the back end using the parameter

Ex:  
This parameter can be from a link  
**http://example.com/app/accountView?id=' or '1'='1**Which would result in the back end returning the entire table instead of by ID, since 1=1 is always true.

### What are the best practices to protect your web application from injection?

The easiest and safest solution to protect your application is to use an API, which avoids the use of the interpreter entirely and provides a parameterized interface.

The API approach can still produce injections if the implementation is bad.

If you do not want to use an API you will need to sanitize all the user inputs, limit the data the user can get from the database, and use server-side input validation.

### What defines an injection secure application?

For an application to be injection-safe inputs and parameters must be sanitized before use.

### Does injection affect my project?

Since I am using an API and I sanitize my inputs/parameters in the logic layer of the application, injection is not a threat to my project.

## Broken Authentication

### Security Risks of broken authentication

Broken authentication is widespread due to the design and implementation of most identity and access controls.

While harder to detect than an injection this flaw can lead to the compromise of one or many accounts.

Depending on the domain of application, this flaw can have very big impacts on your consumers, such as: money theft, identity theft, frauds, or theft of highly sensitive information.

Consumers tend to use the same passwords on multiple services, meaning that if one gets compromised, they might lose other important accounts as well.

### How can your website be attacked using broken authentication?

Most commonly a hacker can attack your application using credential stuffing, brute forcing passwords, or abusing bad session implementation.

### What are the best practices to protect your web application from having a broken authentication?

One of the best possible ways to protect your website is to implement multi-factor authentication to prevent the hacker from accessing the account.

If multi-factor authentication is out of the apps scope, your application should implement weak-password checks, make sure that the password is strong enough by checking length and complexity, make sure your error messaged do not give any type of information to the hacker, delay failed log in attempts, and make sure sessions are implemented is a secure way.

### What defines an authentication secure application?

For an application to be authentication safe it needs to make sure there is not any account data leak or hints in the responses.

Ex:

When a user fails logging it on an existing account, the error message should say “Incorrect Credentials” instead of “Incorrect password”, the same as when he tries to authenticate with a non-existing account.

### How I overcame authentication flaws in my project

Sine my application does not have multi-factor authentication I have decided to use JWT tokens for each user. These provide a little bit more security than sessions since it cannot be abused unless the hacker already has access to the victim’s computer.

I also made sure to not give any hints in the error message sent by the API, as on the Login page it will always return “Invalid Credentials” even if the username is correct. This does not extend to the Register form since a smart hacker would figure out that an error message appears only when a username already exists.

I have made sure that there are no data leaks as the only information the front-end receives on Login is the JWT token and nothing else, all account checks being made in the back end, where the hacker has no access to.

## Sensitive data exposure

### Security risk of data exposure

Data exposure is the most common flaw in web applications. This is due to the fact that a large part of developers do not encrypt the data or the encryption used is weak.  
In the case of a weak encryption the hash used can be easily found using ciphers.

This flaw can lead to the theft of personal information, credentials, personal data and most commonly, credit cards.  
The European General Data Protection Regulation (EU GDPR) requires all European countries to comply with the privacy laws requiring sensitive data to be encrypted using powerful encryption tools.

### How can your app be attacked using data exposure?

If your application does not encrypt their data a hacker can easily grab and used it.

If you re-use weak crypto keys the hacker can gain access to all the information once the crypto key has been broken.

Weak hashing of data can result in theft of data when paired with [injections](#_Injection).

Hackers can also try to steal data from a public connection and steal session data from users, which is commonly kept in cookies. The hacker can then access the user’s session and gain access to their data.

### What are the best practices to avoid data exposure?

Always encrypt sensitive data, and if it is very important data such as credit cards, use the best encryption tools available.,

Do not store sensitive data if not needed. This is the reason most checkouts never remember your credit card info.

Make sure that the encryption tools are the best.

Store passwords and sensitive data using the best hashing functions available.

### What defines an application with secure data?

In a secure application data needs to be encrypted and handled with care, especially sensitive data. Smart hashing and powerful encryption tools are a must have for a secure application,

### How I handle data in my project

In my project there is not any sensitive data other than the account information, for which the password is hashed and the JWT is encrypted.

## Cross-Site Scripting (XSS)

### Security risk of XSS

XSS is the second most common flaw in web applications. It is found in two thirds of all applications.

XSS is a type on injection where malicious scripts are injected into trusted website in the form of a browser side script. The injection can cause simple annoyances or complete account compromise. In the worst-case scenario, an XXS attack can access the session cookie and take over the user’s session, install Trojans or modify the presentation of content (used in credit card fraud).

### How can your app be attacked using XSS?

Reflected XSS: The application or API includes unvalidated and unescaped user input as part of HTML output. A successful attack can allow the attacker to execute arbitrary HTML and JavaScript in the victim’s browser. Typically, the user will need to interact with some malicious link that points to an attacker-controlled page, such as malicious watering hole websites, advertisements, or similar.  
  
Stored XSS: The application or API stores unsanitized user input that is viewed later by another user or an administrator. Stored XSS is often considered a high or critical risk.  
  
DOM XSS: JavaScript frameworks, single-page applications, and APIs that dynamically include attacker-controllable data to a page are vulnerable to DOM XSS. Ideally, the application would not send attacker-controllable data to unsafe JavaScript APIs.

(Credit - <https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)>)

### What are the best practices to avoid XSS? / What defines an application that’s XSS secure?

Using a framework that escaped XSS by design, such as React.

Not using untrusted HTTP requests.

Enable Content Security Policy (CSP)

### Does XSS affect my project?

No, XSS does not affect my project since I am using the React framework.

## How to prevent other security risks

Most security risks can be prevented by following the OWASP security guides. It offers an in depth explanation of how they work and how to prevent them.

# Security risks of a distributed system

## Snooping

Snooping is the process of listening to the network traffic between (in this case) API and WebApp.

This flaw cannot be prevented unless the user connect trough a secure connection.

A user can avoid data snooping by using safe connection; it is not recommended to connect to random networks.

## Denial of Service / Distributed Denial of Service (DoS/DDoS)

DoS is a malicious attempt to disrupt the normal traffic of the targeted server, service, or network by overwhelming the target with a flood of Internet traffic in the form of packets.

Popular websites such as Facebook and YouTube implement algorithms to detect such attacks and ignore the data flood.

## Spoofing

A Spoofing attack in a situation where a hacker successfully identifies as a user by falsifying data, to get their data.

Spoofing can easily be avoided using 3rd party software or implementing a packet filtering algorithm.

# General good practices

1. Be aware of the application’s flaws.

Testing for flaws is a must if you want to deploy a serious application, and you need to be aware of how hackers would attack your system. Try to patch out as many flaws as possible.

In case you get attacked, think about why and how someone might want to access your system/data and try to patch up every hole in your application.  
  
YOU SHOULD:

* Know and understand how your application and API interact with external entities.
* Determine the severity of the attack and come up with a defense plan.
* Determine the security risk for the attack using a risk assessment model (such as DREAD).
* Deploy countermeasures trying to protect against the highest risks to your application.

1. Protect your API.

APIs usually contain direct access to your most important data, so getting your API injected or manipulated means a total leak of data.

Some simple steps to make your API more secure:

* Remove unused code.
* Encrypt data and Hash important data.
* Remove outdated libraries/ libraries that you no longer use.
* Keep your salts and peppers in separate files that are not public.
* Change your encryption keys regularly.

1. Protect your code from injection.

Sanitize Everything!  
As mentioned in the [Injection](#_Injection), most data leaks happen because user input is not sanitized. It is highly recommended that you sanitize all user input before using it. Avoid using user input without sanitizing it first.

1. Keep everything updated.

It is very common to use libraries and packages in these times, and these can have their own flaws; the more you have the higher the change of a major flaw. It is recommended to keep them updated to the latest version.

HEADS UP: If the library/package is open source and you are aware of a flaw you can fix it yourself after informing the developers.

# General bad practices

* Not sanitizing user inputs.
* Using outdated/dropped libraries.
* Storing important data such as credit cards (this is illegal), personal data, etc.
* Sending personal data over e-mail.
* Keeping logs in the application.
* Not being aware of the flaws before deployment.
* Having unused code that could be abused.

# Recommendations

1. Always test for flaws in your application

There are many tools online to test for all the flaws that exist.

1. Be extra careful with sensitive data.

Sensitive data is no joke, you should always take the steps to protect it. Use encryption, hashing and algorithms to make sure that no one can access it.

1. Know how to defend against attacks.

No application is perfect, however that is no excuse to not know how to prepare and defend against an attack. You should read up and be aware about how to defend your application from any type of attack.

1. Monitor your application.

Keep logs and actively monitor your deployed application to make sure that no attack is happening.

Logging is also helpful for application that involve monetary transactions.

# References

* <https://www.otago.ac.nz/distance/otago021395.pdf>
* <https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf>
* <https://www.veracode.com/security/spoofing-attack>
* <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>
* <https://en.wikipedia.org/wiki/Threat_model>
* <https://appdevelopermagazine.com/security-first:-5-tips-for-building-a-secure-mobile-app-from-the-ground-up/>
* <https://owasp.org/www-community/attacks/xss/>
* <http://www.vsicam.com/_faq/what-is-igmp-querying-and-igmp-snooping-and-why-would-i-need-it-on-my-network/>
* <https://www.researchgate.net/publication/309285125_Security_Issues_in_Distributed_Computing_System_Models>