OWASP Security

Carlo van Kessel
S-DB-IP3 and S-DB-GP3
S3-DB03
Hans van Heumen, Marc van Grootel
13-01-2023
V3

Version control

Version	Changes
1	Initial version.
2	Added How do you prevent Broken Access Control?
3	Added What did I do to prevent this?

Contents

DOT Research Framework

For this research report I use the DOT framework. This research framework is split in three main parts:

1. The "What" of my research (the domains)
2. The "Why" of my research (the trade-offs)
3. The "How" of my research (the strategies and methods)

For more information about the DOT framework:
https://ictresearchmethods.nl/The_DOT_Framework

“What” is security in software?

Software security is the practice of ensuring the confidentiality, integrity, and availability of computer systems and data by designing and implementing security controls and measures. This includes protecting against various types of attacks, such as hacking, viruses, and malware, as well as implementing security best practices and protocols.

A foundation called OWASP creates a top 10 every three years about the most common security mistakes that still occur and how to prevent these mistakes. The last top 10 that was made was in 2021:

1. A01:2021-Broken Access Control

Broken access control means that there is no control over who has access to certain information or functionalities. This then can lead to data leaks, where user data may be in the wrong persons hands.

For example: A user can’t access other users’ information, this should be blocked with a unauthorized notification.

2. A02:2021-Cryptographic Failures

Cryptographic failures means that data should be extra protected, such as passwords, credit card numbers, health records, personal information, and business secrets. If this is not secured hackers can take this information and use/sell it.

3. A03:2021-Injection

Injection is a way of hacking a Web application. This can be done in a variety of ways. Consider SQL injection, where you can execute an SQL query on the Web application's database. Then you can access information encrypted or maybe not, this is of course not good.

4. A04:2021-Insecure Design

Insecure design is about thinking ahead before you start building you application. If you have a insecure design you cant fix it while building the application, you first need a secure design. Think about who can access the database or who can access the API.

5. A05:2021-Security Misconfiguration

With security misconfiguration you need to think about using old software or it is vulnerable. Also, you can think about having useless components or unnecessary libraries installed.

6. A06:2021-Vulnerable and Outdated Components

This involves the use of outdated software. Think about using an outdated version of a library or an older operating system. It is important that you are always up to date with the software you are using. The moment you use outdated software, you may have a vulnerability. It is also important to know what components you are using. And what the components do. The moment you use a component you don't know; you can have a vulnerability.

7. A07:2021-Identification and Authentication Failures

This involves identifying a user. Think about not allowing a weak or easy password. It is important to think about what you allow and don't allow for users to use in your application.

8. A08:2021-Software and Data Integrity Failures

This has to do with what kind of infrastructures you are using. Consider using a libary you don't know where it came from or how it works. Or an improper CI/CD pipeline. It's important to know what you're using and how it works.

9. A09:2021-Security Logging and Monitoring Failures

This is about logging a web application. It is important to know what is happening on your Web application. The moment you don't know what is happening, you can't know if there is a vulnerability. This can be done with logging in specific parts of the application.

10. A10:2021-Server-Side Request Forgery

These issues happen when a web app retrieves information from an external source without checking the URL input by the user. This vulnerability allows an attacker to manipulate the app into sending custom requests to unintended locations. With the increasing popularity of cloud services, this type of error has become a significant concern.

"Why" do you need security?

Security in software is necessary to protect sensitive information and prevent unauthorized access, modification, or destruction of data. It also helps to protect systems and networks from malicious attacks, such as viruses and malware, and ensures the confidentiality, integrity, and availability of information. Additionally, security in software can help to comply with regulatory requirements and industry standards and can help to prevent financial losses and reputational damage.

"How" do you prevent Broken Access Control?

The first step in preventing broken access control is to properly design and implement your application's authentication and authorization systems. This means ensuring that users are properly authenticated and authorized before they are allowed access to certain resources or functionality. This can be done by implementing a role-based access control (RBAC) system, where different users are assigned different roles that determine what they are allowed to access. Additionally, it is important to ensure that your authentication and authorization systems are secure and resistant to common attacks such as SQL injection and cross-site scripting (XSS).

Another key step in preventing broken access control is to properly validate user input. This means ensuring that any input from users is properly sanitized and validated before it is used in any way. This can include checking that the input is in the correct format and that it does not contain any malicious code or SQL injection attacks. Additionally, it is important to ensure that your application is properly configured to handle invalid input, such as by displaying an error message or redirecting the user to a different page.

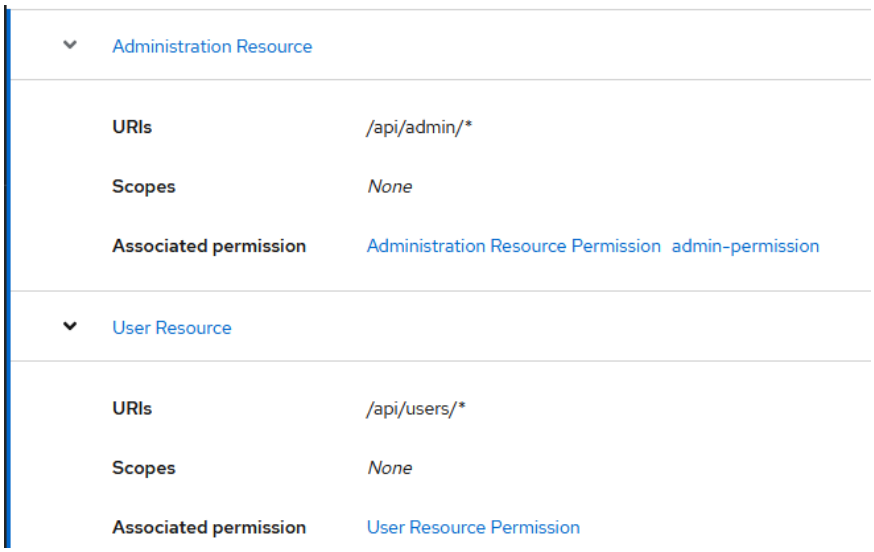
Another important aspect of preventing broken access control is to properly secure your application's data and resources. This means ensuring that sensitive data is properly encrypted and that access to it is restricted to only those who need it. Additionally, it is important to ensure that your application's resources are properly protected from unauthorized access, such as by using firewalls and intrusion detection systems.

Finally, to prevent broken access control, it is important to regularly test and monitor your application for vulnerabilities. This can include performing regular penetration testing to identify any potential vulnerabilities and addressing them as quickly as possible. Additionally, it is important to regularly monitor your application's logs and other security-related data to identify any suspicious activity or potential threats.

In conclusion, preventing broken access control is an essential aspect of securing your application. By properly designing and implementing your application's authentication and authorization systems, validating user input, securing your application's data and resources, and regularly testing and monitoring your application, you can greatly reduce the risk of broken access control and ensure the security of your application and its users.

What did I do to prevent this?

For my single sign on I used Keycloak, here I can easy track and update user permissions. This is done with the endpoint that the user is visiting.



Then in my backend I used a check to see if the user matches the JWT token it sends like this:

```
if (!userService.checkIfUserExists(userEmail) ||
!securityIdentity.getPrincipal().getName().equals(userEmail)) {
    throw new IllegalArgumentException(INVALIDUSER);
}
```

Here in the second if statement I check if the getName equals the user email that is send by the endpoint. So when a different user tries to access this endpoint it fails.

Sources

https://owasp.org/Top10/
https://www.geeksforgeeks.org/how-to-prevent-broken-access-control/