
# File Inclusion Vulnerability & Attack

Zachary Kein - 220277143

## Contents

# File Inclusion Vulnerability

A file inclusion vulnerability affects web applications when users are able to submit input into files or are able to upload files to the server. This vulnerability then allows attackers to read and execute files on the victims server, or even execute code that is hosted on the attackers machine. It is important to test for this vulnerability as you do not want attackers to be able to access hidden files on your server. This allows for password and file stealing, as well as website defacement.

# File Inclusion Attack

There are two forms of file inclusion attacks that exploit this vulnerability. They are Local File Inclusion (LFI) and Remote File Inclusion (RFI). LFI exploits this vulnerability to include and display files that are already located on the local server that is hosting the webapp such as sensitive data or config files. RFI exploits the vulnerability by executing files from a remote server, such as executing malicious code to gain control over the website.

This security test will see whether either LFI or RFI is possible by analysing the content of the MOP website and seeing what information we can gather. Then we will decide whether this attack is possible and see if LFI or RFI would cause greater damage to the website.

# Setting up the Test

To conduct this test, we used the following tools:

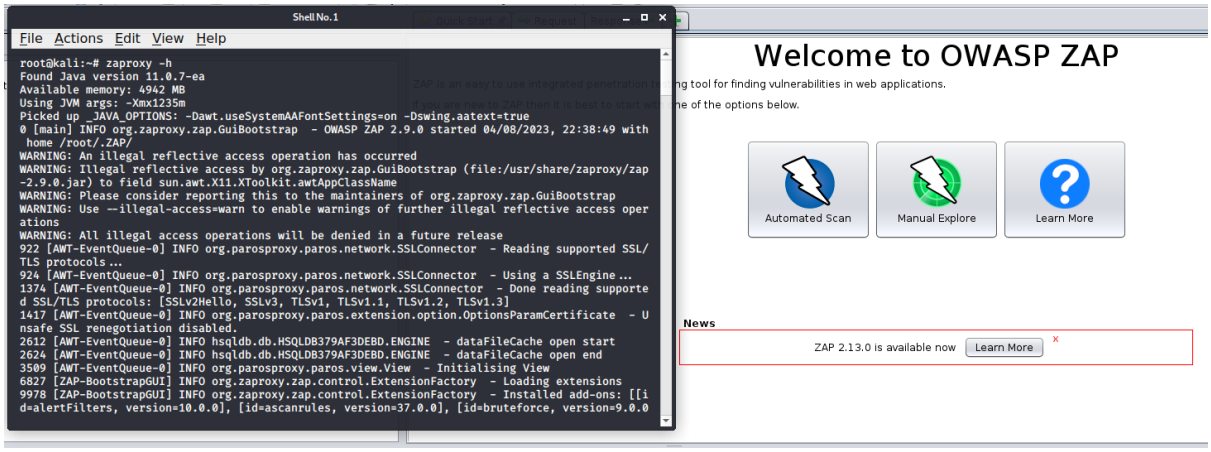
- Kali Linux
- OWASP ZAP

There are a variety of tools that can be used to test the file inclusion vulnerability such as burp suite, but for this test I have chosen OWASP ZAP.

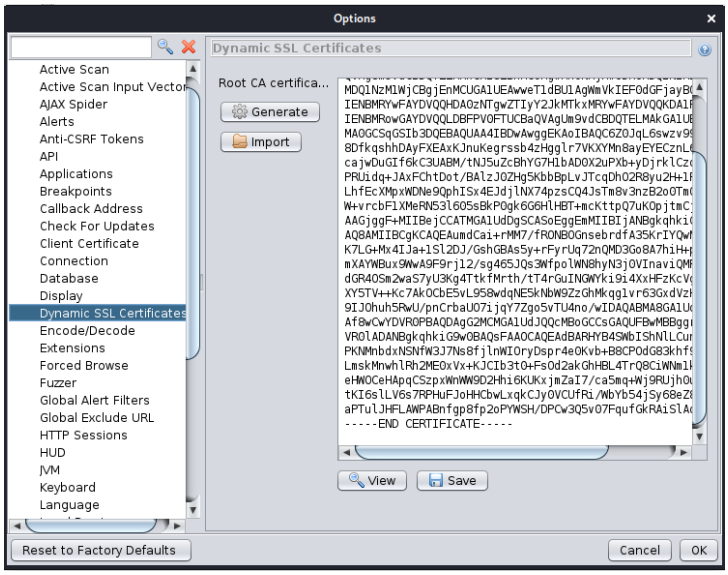
# Testing Method

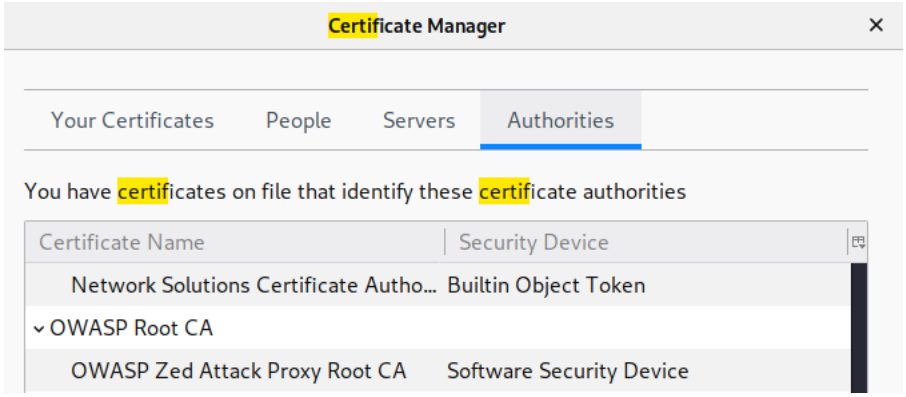
## Launching OWASP ZAP

Using the command ‘zaproxy -h’ launches OWASP ZAP and takes us to the program

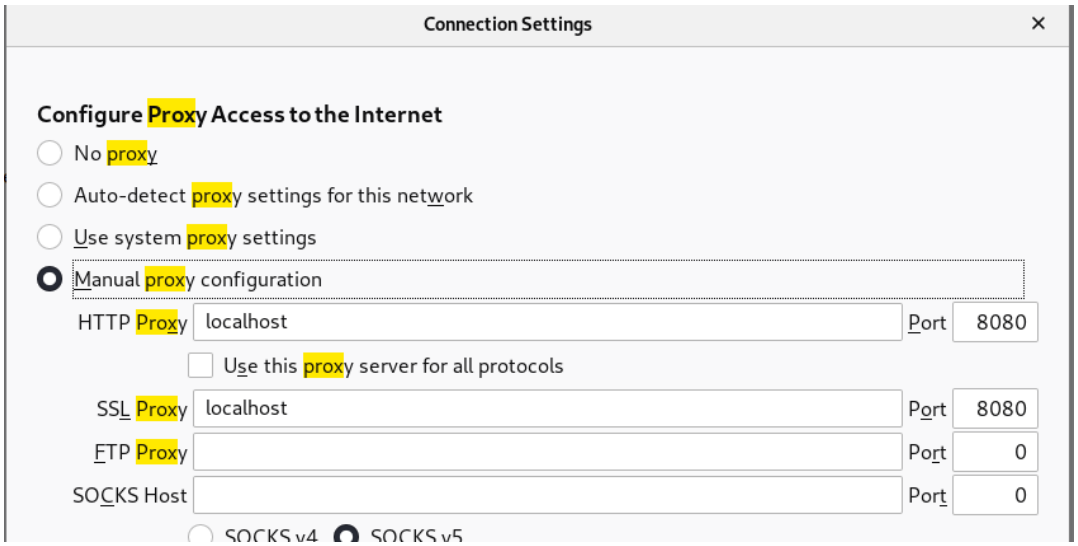


It is important to generate a new Root CA Certificate and import it into your browser, otherwise the browser will block websites due to the malicious activity we are performing.



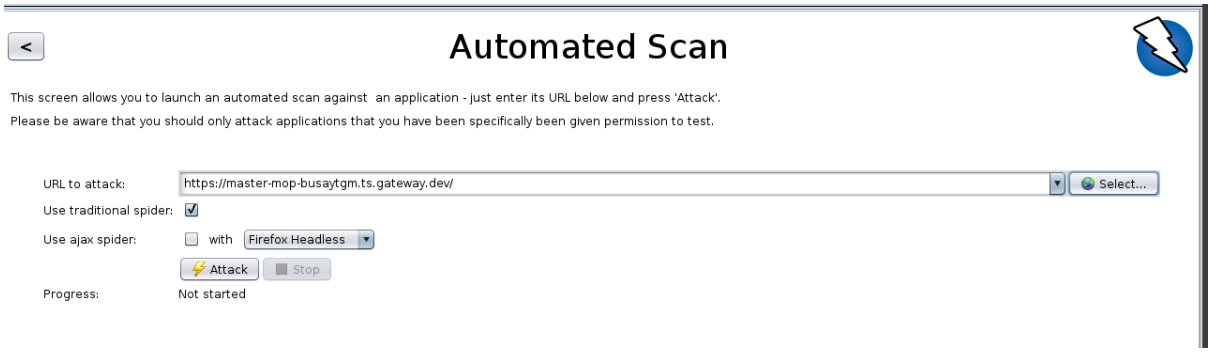




I have also added OWASP ZAP as a proxy network to my browser so that it is able to read my browsing data and gather information.

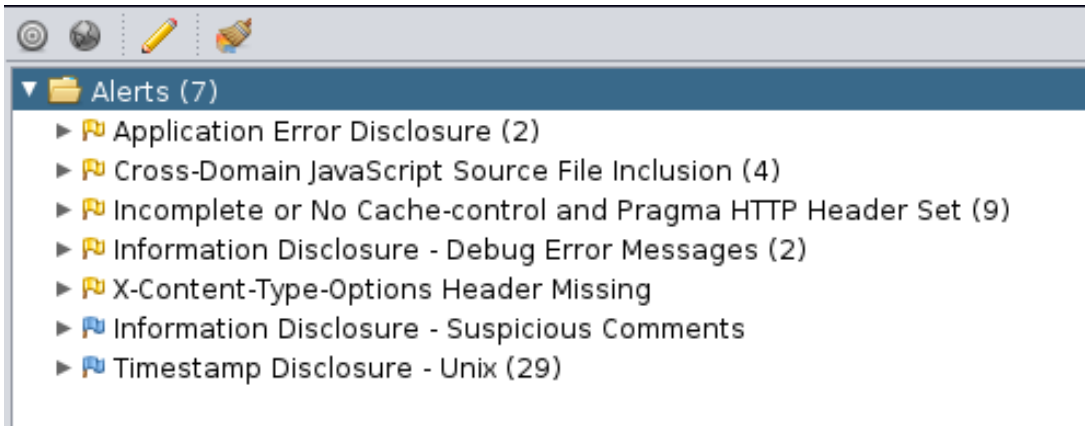


## Completing a Scan

We will first complete a scan on the MOP website to reveal any vulnerabilities present as it will scan each webpage of the website.

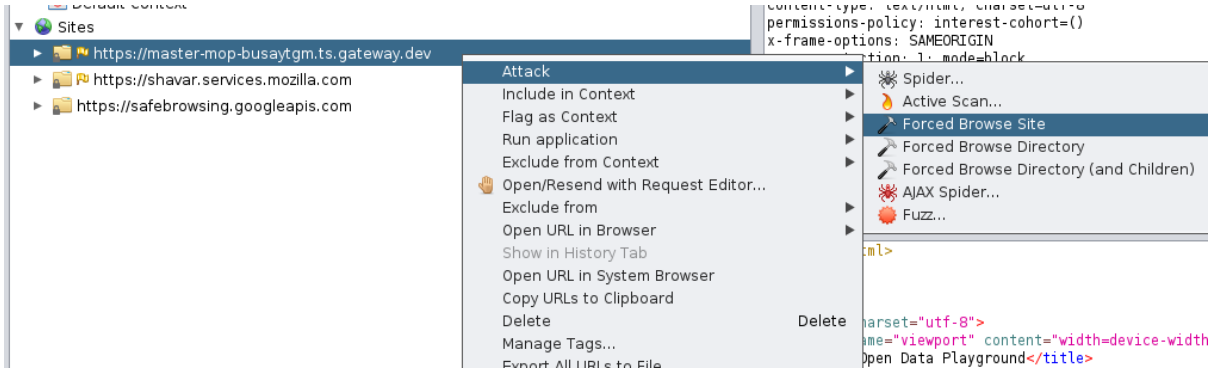



The results of the scan:

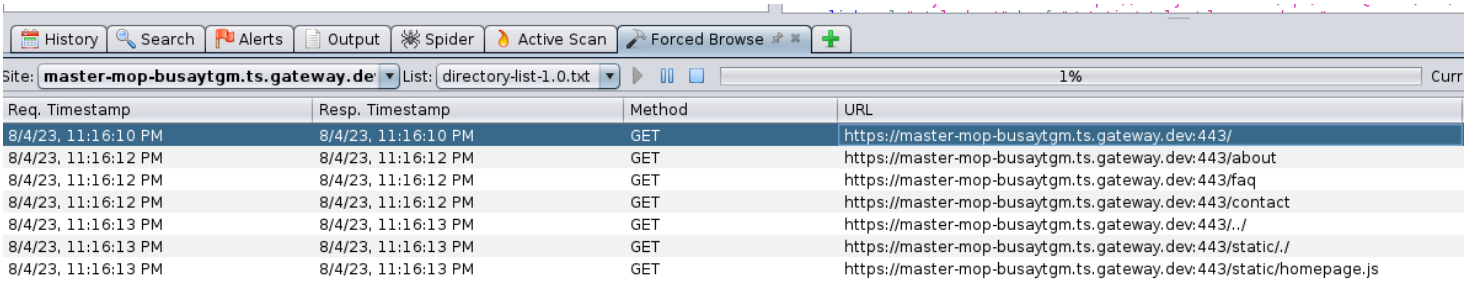



## Forced Browse

We can then do a forced browse on the MOP website to reveal every webpage as there may be hidden parts of the website with files that could potentially be exploited:



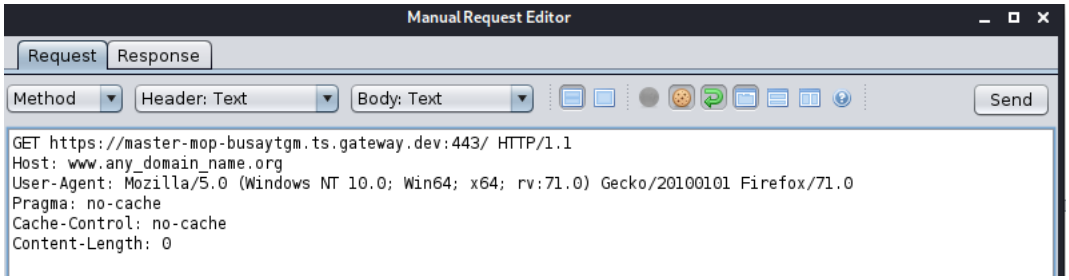
Results from Forced Browse:

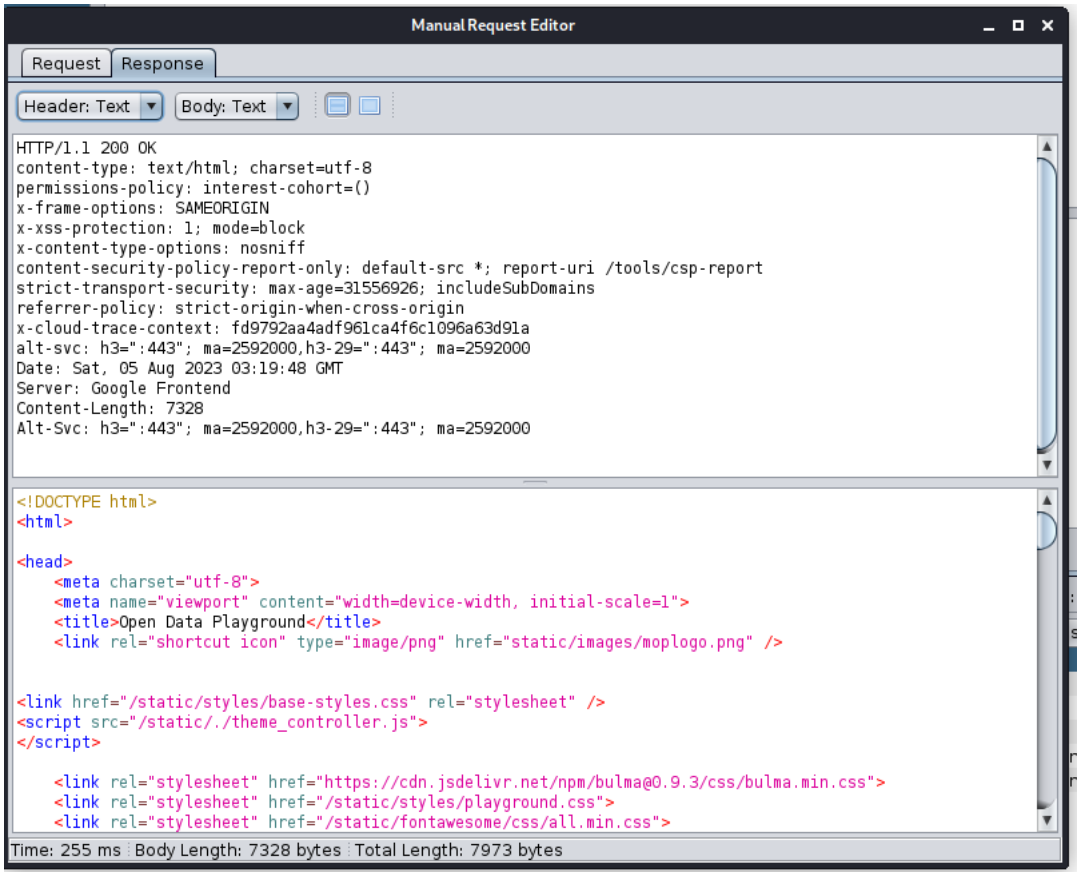



| Req. Timestamp | Resp. Timestamp | Method | URL |
|---|---|---|---|
| 8/4/23, 11:16:10 PM | 8/4/23, 11:16:10 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/ |
| 8/4/23, 11:16:12 PM | 8/4/23, 11:16:12 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/about |
| 8/4/23, 11:16:12 PM | 8/4/23, 11:16:12 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/faq |
| 8/4/23, 11:16:12 PM | 8/4/23, 11:16:12 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/contact |
| 8/4/23, 11:16:13 PM | 8/4/23, 11:16:13 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/../ |
| 8/4/23, 11:16:13 PM | 8/4/23, 11:16:13 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/static/./ |
| 8/4/23, 11:16:13 PM | 8/4/23, 11:16:13 PM | GET | https://master-mop-busaytgm.ts.gateway.dev:443/static/homepage.js |

## Finding Vulnerabilities

Finally, with these results we can use the manual request editor to reveal the contents of the webpage and look for any potential file inclusion vulnerabilities that can be exploited.



## Results

After conducting the security check on the MOP website and searching for any potential ways to conduct a file inclusion attack, the results show that due to the lack of user submitted data and sensitive data on the website, a file inclusion attack should not be something of worry for the MOP website. As the website evolves and updates, there will eventually be some user input and sensitive data that will be stored on the MOP website so it will be important to constantly check after updates that sensitive data is properly secured as any hidden data can be found through tactics such as these.

File inclusion attacks can result in all data from a website being fully compromised. Therefore, it is extremely important to begin to put into place the structures that will help mitigate any of these attacks. Some steps that I would recommend taking to protect the MOP website from these attacks are the following:

1. Input validation and sanitization

Any user input should be validated and sanitized before it is used to include files or perform any other actions. It will also be important to use whitelists that specify any allowed values or patterns for input.

2. Avoiding dynamic file inclusion

User supplied input that is directly in the file inclusion functions should be avoided. Alternatives to dynamic inclusion should be considered such as using a switch statement.

3. Smart file location placement

All sensitive and critical data should be stored outside of the web root directory to prevent direct access through the browser.

4. Using well-established frameworks and libraries

Well-established frameworks and libraries should be used as they typically have built-in security mechanisms that work against common vulnerabilities, which includes file inclusion attacks

5. Input whitelisting

A strict whitelist of allowed files and directories should be in place if user input is used for file inclusion.

6. Directory access control

The web server should be configured to restrict access to sensitive directories and files. Proper permissions must be in place so that only authorized users can access sensitive resources.

7. Regular penetration testing

With each iteration of the website, regular penetration testing should become routine as to test for common infiltration tactics.