What this report contains

Executive Summary

This section provides a non-technical overview of the scope of the project, the key findings from the testing work carried out and highlights the areas we see as most important or urgent in needing attention to improve the security of the system under test.

Target Information

This lists what was tested and the type of testing carried out.

Methodology

This is a short, technical description, of physical way that the testing was undertaken.

Issue Summary

This is a high level, technical, summary of each issue found. It's common for this section to contain multiple examples of the same type of issue as we list every instance of an issue we find so that there is a clear list of everything that needs to be fixed – i.e. If we find a form that is insecure it's possible that each individual field within the form has the same type of issue and so if the form has five fields; First name, Last name, Email address, Company name and comments these would be listed as 5 issues as each field needs attention.

Vulnerability by Severity

This is a visual representation of the different levels of issues found to provide a quick overview of the number of each type individually and in relation to the application as a whole.

Details of Individual Issues

This is a detailed list of each issue found containing, where possible:

- Affected assets – what in the system has the issues
- Severity – on a 5-point scale from 'Critical' to 'For information only'
- OWASP / CWE mapping – how the issue relates to security testing methodologies or databases
- Technical description – a detailed technical explanation of the issue
- Steps to reproduce – how to verify the issue exists and to check that it has been fixed
- Business impact – a non-technical description of the potential impact of the issue
- Solution and/or mitigation – what's required to fix the issue
- External technical reference links – additional information in the public domain to provide more insight into the type of issue, other ways to fix it etc.

As with the issue summary, every instance of an issue is listed individually to provide a detailed, step-by-step guide for all fixes required.

Executive Summary

Appsecco was contracted by Law Firm Services to conduct Web Application Security Testing to determine if there were security weaknesses in the LFS Gateway application, the implementation of the Identity Server, the backend LFS API server and the tenant applications that can render the environment insecure and allow an attacker to gain access to any data that is accessible via them or gain access to the underlying operating system.

The testing was carried out between 7th May to 11th May 2018 on the testing setup of the LFS Identiy Server, LFS Gateway and the associated API servers and web applications

The OWASP Top-10 2013 and CWE was used as the reference frameworks to evaluate and categorise the discovered security issues.

Approach

Our testing approach entailed using a demo implementation of the environment with two tenants created on the server and activated by following an email containing an Opt-In link. The application environment uses API endpoints to authorize, retrieve and update data while a standard Angular based web application is used to update the UI retrieved by backend JS requests.

Once the setup was completed and access obtained, the following testing approach was used along with the standard approach to testing for the OWASP Top 10 2013 and API testing:

1. The opt-in link was analyzed to check if it can be re-used, tampered with to activate other users, access a different account and to check if it uses non-standard communication channels
2. The application was tested for cross tenant access using tampered tokens, tampered host headers and tampering of various GET and POST parameters
3. As the environment uses an Identity Server for the authentication and authorization, an attempt was made to check for token expiration and revocation and usage after the token is supposedly not valid
4. The access tokens and other JWT tokens were subjected to a manual revocation to check if the token continued to provide access
5. The application’s logout functionality was tested for both the cookie session being destroyed and for the tokens to be revoked
6. The forgot password was checked and an attempt was made to tamper and change the password of a different tenant
7. Various parameters were tested for injection attacks in an attempt to invoke errors or cause user supplied data to mix with a backend query
8. An attempt was made to perform authorization bypass by reusing expired tokens, removing tokens completely, tampering tokens using tokens belonging to different account
9. The application was also subjected to various input that could be used to potentially leak information to the client.
10. The JavaScript was reviewed in an attempt to discover secrets, tokens, keys and hardcoded variables that may cause information to be disclosed to the client
11. The JWT tokens were decoded, an offline brute force was attempted to verify if weak passwords are used to obtain a verifiable signature so that tampered tokens could be sent to the server.

12. An attempt was made to discover possible hidden API endpoints that are not part of the documentation that was shared
13. The provided documentation was used as a reference to provide various test input to the API endpoints and create Activities, scope and tokens that were tampered interchangeably to detect authorization issues, token reuse, weak scopes and access.

Summary of Results

The environment was tested extensively for any authentication and authorization weaknesses that could allow an attacker to access data across tenant boundaries or gain access to a user's account by any other vulnerabilities prevalent in the applications in scope.

The testing showed that the environment is vulnerable to weaknesses that could allow a potential attacker to perform Man-in-the-Middle attacks or reuse access tokens that survive a revocation attempt. This could allow an attacker access to the data that the applications work with and may result in lowered trust with the environment.

No Critical vulnerabilities were identified. The following is a quick summary of the issues discovered during the assessment:

- The URL, that would allow a user to opt-in to an Activity or setup their account at the very beginning of the user lifecycle, makes requests over HTTP instead of a secure HTTPS channel
- The access tokens are not revoked even after an explicit request is made to the Identity Server's revocation endpoint
- The application's logout functionality is not completely implemented as the access tokens are not expired or revoked on the server while only the cookies are unset allowing a user to access data via the API endpoints instead of the application even after a successful logout
- A JavaScript library used in the environment is of an older version that has been shown to be vulnerable to a Cross Site Scripting attack. Given the complexity of the exploit and the conditions under which this becomes dangerous, this issue has been rated low.

Conclusion

The application's environment was found to withstand injection and parameter modification attacks. The backend is built with authentication and authorisation in mind that prevented any cross tenant data access to become possible. This was tested by manipulating the tokens, cookies as well as the host headers that the API's use to send data back. The application exhibits sturdy access control and allows only valid and authorised users to perform actions.

However, the application design does not properly take care of session expiration issues which allows a user to continue having access to the API backend using requests outside the web application where the user is logged out. Also, the revocation endpoint which according the documentation for Identity Server should be utilised to retire a token is not used. This allows an attacker to access restricted content even after a forced revoke is invoked. Finally, the application executes the Opt-In functionality with a link that exchanges data using an insecure communication protocol.

The discovered issues have created an environment that is potentially vulnerable to attacks that could lead to a loss in data and unauthorised access to sensitive information. These issues need to be fixed to give assurance to the users of this environment.