*XSS Exploitation Attack Tree*

User unintentionally runs further scripts which give further access to the attacker.

User has now unintentionally given access to a remote shell.

Attacker could potentially view stored credentials.

User is sent malicious scripts coming from the attacker’s server

Remote shell initiated via the attacker’s server

Users session is relayed to the attacker’s server

User accesses the vulnerable page with the injected payload attached.

Malicious payload injected into the field.

Vulnerable field found within application.

Compromised

**STRIDE *Threat Modelling***

Spoofing:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Spoofing a role | Declares themselves as being that role | Opening an account different to that of the | Implement a verification system to ensure only users who require a certain role have access to one. |
| Spoofing a machine | ARP Spoofing, IP spoofing, DNS spoofing |  |  |
| Spoofing a process | Creates a secondary login page different to that of the native app. | Screen grabbing and spoofing a fake login page to gain credentials. | Implement |

Tampering:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Tampering the login request | Tampering with the request to authenticate into the web app, could potentially allow privilege escalation. | Setting the user session token to that of an admin. | Ensure that the session reauthenticates upon logging in with that of the requested credentials. |
| Tampering with the page responses | Inserting arbitrary code to understand how the page handles the response. | Inserting simple XSS alerts to see how the page handles it. <script>Hello;</script> | Implement sanitation of inputs |

Repudiation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Repudiating an action | Claims not to have logged in. | It couldn’t have been me, I didn’t login with that account! | Increase logging to cover logged in sessions |
| Claims not to have received | I never received that notification/email! | Increase logging to cover emails/notifications/actions sent |

Information Disclosure:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Extracts user data | Exploits vulnerabilities such as SQL injection to read DB tables | SELECT \* FROM Users WHERE UserID = 105 OR 1=1; | Sanitation on user inputs to ensure SQL injections do not take place. |
| Extracts server information | Reads the server response headers for versions of services being used. | HTTP Response Header, X powered by messages. | Restrict the information relayed in the response headers |

Denial of Service:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Repeated invalid login attempts | Automated login attempts using invalid credentials to slow down the server. | Brute force using invalid creds **could** cause performance issues. | Implement login attempts per IP or MAC. |
| Slow lloris attack | Sends partial http requests to the web server, but none ever being completed. As a result, the webserver will open more connections assuming at some point the requests will be completed. | Slow lloris is an application layer DDoS attack. | Increase the number of clients the server will allow. Limit the number of connections a single IP address can attempt. |

Elevation of Privileges:

|  |  |  |  |
| --- | --- | --- | --- |
| **Threat Example** | **What the attacker does** | **Notes / Example** | **Mitigation / Countermeasures** |
| Download of sensitive data | Enumerates a download link to expand the scope for download. | Changing the download parameter to something different could allow download of different files. | Perform additional checks upon download of files to ensure only the user with the correct permissions is authorised. |
| Inject a command | Can inject a command that the system will run a higher privilege level. |  | sanitise user inputs allowing only certain types of characters and input. |
| Spoofing a user | Source co |  |  |

**Threat model *Data Flow Diagram***

