|  |  |
| --- | --- |
| **Observation No. 1** | |
| **Vulnerability** | CSRF Attack |
| **Status** | Open |
| **Severity** | **Critical** |
| **Description** | Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. |
| **Remediation** | Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. |
| **Affected URLs** | * http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 * http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 |
| **References** | Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. |
| **Proof Of Concept** | Screenshot shared below. |
| Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user. | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |

|  |  |
| --- | --- |
| **Observation No. 2** | |
| **Vulnerability** | SQL Injection |
| **Status** | Close |
| **Severity** | **High** |
| **Description** | SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input. |
| **Remediation** | SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input. |
| **Affected URLs** | * http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 * http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 |
| **References** | SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input. |
| **Proof Of Concept** | Screenshot shared below. |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |

|  |  |
| --- | --- |
| **Observation No. 3** | |
| **Vulnerability** | XSS |
| **Status** | Open |
| **Severity** | **Medium** |
| **Description** | Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. |
| **Remediation** | The primary defenses against XSS are described in the OWASP XSS Prevention Cheat Sheet. Also, it’s crucial that you turn off HTTP TRACE support on all web servers. An attacker can steal cookie data via Javascript even when document.cookie is disabled or not supported by the client. This attack is mounted when a user posts a malicious script to a forum so when another user clicks the link, an asynchronous HTTP Trace call is triggered which collects the user’s cookie information from the server, and then sends it over to another malicious server that collects the cookie information so the attacker can mount a session hijack attack. This is easily mitigated by removing support for HTTP TRACE on all web servers. |
| **Affected URLs** | * http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 |
| **References** | https://owasp.org/www-community/attacks/xss/ |
| **Proof Of Concept** | Screenshot shared below. |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |
| http://localhost:3000/observation/create/64f96fb85eeaaba1315bdbb7 | |