

|  |  |
| --- | --- |
| **Project Name** | DVWA |
| **Tested App Version** | 1.0.0 |
| **Start & End Dates** | 04/01/2019 - 04/26/2019 |
| **Analyst Name** | Prasanth |

**SUMMARY OF FINDINGS**

|  |  |
| --- | --- |
| **Risk** | **Count** |
| Critical | 0 |
| High | 3 |
| Medium | 6 |
| Low | 0 |
| Total | 9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vulnerability or Condition Type** | | **Risk** | **Severity** | **Retest Status** |
| **1** | **Weak Session ID** | **Medium** | Medium |  |
| **2** | **WEAK PASSWORD POLICY** | **Medium** | Medium |  |
| **3** | **Autocomplete on sensitive fields** | **Medium** | Medium |  |
| **4** | **MISSING SECURITY HEADERS** | **Medium** | Medium |  |
| **5** | **SQL INJECTION** | **High** | High |  |
| **6** | **Cross site scripting – reflected** | **High** | High |  |
| **7** | **Direct Dynamic Code Execution - Eval Injection** | **High** | High |  |
| **8** | **HTTP VERB TAMPERING** | **Medium** | Medium |  |
| **9** | **SENSITIVE INFORMATION IN RESPONSE HEADERS** | **Medium** | Medium |  |

**DETAILED FINDINGS**  
**Finding #1: Weak Session ID**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 4.3 |
| **Vector** |
|  |
| **Occurrences** |
| 1 |
| **Affected Components** |
|  |
| **Details** |
| Session identifiers should be at least 128 bits long to prevent brute-force session guessing attacks.A shorter session identifier leaves the application open to brute-force session guessing attacks. If an attacker can guess an authenticated user's session identifier, he can take over the user's session. In addition, the attacker can implement a brute force technique to generate and test different values of session ID until he successfully gets access to the application. |
| **Suggested Remediation** |
| Session identifiers should be at least 128 bits long to prevent brute-force session guessing attacks. |

**Proof of Exploit**  
No Attachments  
  
**Finding #2: WEAK PASSWORD POLICY**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 5.6 |
| **Vector** |
|  |
| **Occurrences** |
| 3 |
| **Affected Components** |
|  |
| **Details** |
| The most prevalent and most easily administered authentication mechanism is a static password. The password represents the keys to the kingdom, but is often subverted by users in the name of usability. In each of the recent high profile hacks that have revealed user credentials, it is lamented that most common passwords are still: 123456, password and qwerty. |
| **Suggested Remediation** |
| Introduce additional authentication controls (i.e. two-factor authentication) Minimum and maximum length Require mixed character sets (alpha, numeric, special, mixed case); Do not contain user name; No password reuse. |

**Proof of Exploit**  
No Attachments  
  
**Finding #3: Autocomplete on sensitive fields**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 3.1 |
| **Vector** |
|  |
| **Occurrences** |
| 3 |
| **Affected Components** |
|  |
| **Details** |
| Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application. The stored credentials can be captured by an attacker who gains control over the user's computer. |
| **Suggested Remediation** |
| To prevent browsers from storing credentials entered into Login form, include the attribute autocomplete = `off` in username, password and sensitive fields. |

**Proof of Exploit**  
No Attachments  
  
**Finding #4: MISSING SECURITY HEADERS**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 5.0 |
| **Vector** |
|  |
| **Occurrences** |
| 25 |
| **Affected Components** |
|  |
| **Details** |
| Content Security Policy is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting XSS and data injection attacks. These attacks are used for everything from data theft to site defacement to distribution of malware. Use this header to enable browser built-in XSS Filter. It prevent cross-site scripting attacks. X-Frame-Options response header improve the protection of web applications against Clickjacking. It declares a policy communicated from a host to the client browser on whether the browser must not display the transmitted content in frames of other web pages. Setting this header will prevent the browser from interpreting files as something else than declared by the content type in the HTTP headers. X-Content-Type-Options: nosniff If your server returns X-Content-Type-Options is nosniff in the response, the browser will refuse to load the styles and scripts in case they have an incorrect MIME-type The Access-Control-Allow-Method is a response header used by the server to describe the methods the clients are allowed to use. HTTP Strict Transport Security (HSTS) is an opt-in security enhancement that is specified by a web application through the use of a special response header. Once a supported browser receives this header that browser will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS. It also prevents HTTPS click through prompts on browsers. HTTP Public Key Pinning (HPKP) is a security mechanism which allows HTTPS websites to resist impersonation by attackers using mis-issued or otherwise fraudulent certificates. (For example, sometimes attackers can compromise certificate authorities, and then can mis-issue certificates for a web origin.). The HTTPS web server serves a list of public key hashes, and on subsequent connections clients expect that server to use one or more of those public keys in its certificate chain. Deploying HPKP safely will require operational and organizational maturity due to the risk that hosts may make themselves unavailable by pinning to a set of public key hashes that becomes invalid. With care, host operators can greatly reduce the risk of man-in-the-middle (MITM) attacks and other false authentication problems for their users without incurring undue risk. |
| **Suggested Remediation** |
| Content Security Policy : script-src 'self ' Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page X-XSS-Protection : X-XSS-Protection: 1; mode=block Enables the XSS Filter. Rather than sanitize the page, when a XSS attack is detected, the browser will prevent rendering of the page. X-Frame-Options: deny Browser refuses to display requested document in X-Content-Type-Options: nosniff If your server returns X-Content-Type-Options is nosniff in the response, the browser will refuse to load the styles and scripts in case they have an incorrect MIME-type Access-Control-Allow-Credentials : true The Access-Control-Allow-Credentials response header indicates whether or not the response to the request can be exposed to the page. It can be exposed when the true value is returned. Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD Strict-Transport-Security : max-age=31536000; includeSubDomains; preload The 'preload' flag indicates the site owner's consent to have their domain preloaded. The site owner still needs to then go and submit the domain to the list. Public-Key-Pins : pin-sha256='d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM='; pin-sha256='E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g='; report-uri='http://example.com/pkp-report'; max-age=10000; includeSubDomains |

**Proof of Exploit**  
No Attachments  
  
**Finding #5: SQL INJECTION**

|  |
| --- |
| **Risk** |
| High |
| **Severity** |
| High |
| **CVSS Base Score** |
| 7.4 |
| **Vector** |
|  |
| **Occurrences** |
| 2 |
| **Affected Components** |
|  |
| **Details** |
| SQL injection is a code injection technique that might destroy your database and it is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input. |
| **Suggested Remediation** |
| The use of prepared statements with variable binding and stored procedures are not always safe from SQL injection. |

**Proof of Exploit**  
No Attachments  
  
**Finding #6: Cross site scripting – reflected**

|  |
| --- |
| **Risk** |
| High |
| **Severity** |
| High |
| **CVSS Base Score** |
| 7.1 |
| **Vector** |
|  |
| **Occurrences** |
| 2 |
| **Affected Components** |
|  |
| **Details** |
| Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. 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. |
| **Suggested Remediation** |
| Never Insert Untrusted Data Except in Allowed Locations HTML Escaped Before Inserting Untrusted Data into HTML Element Content Attributed Escape Before Inserting untrusted data into HTML Common Attributes JavaScript Escape Before Inserting Untrusted data into javascript data values Css escape and strictly validate before inserting untrusted data into html style property values |

**Proof of Exploit**  
No Attachments  
  
**Finding #7: Direct Dynamic Code Execution - Eval Injection**

|  |
| --- |
| **Risk** |
| High |
| **Severity** |
| High |
| **CVSS Base Score** |
| 7.1 |
| **Vector** |
|  |
| **Occurrences** |
| 2 |
| **Affected Components** |
|  |
| **Details** |
| The eval function enables dynamic execution of JavaScript source code.If unvalidated input passed through this eval function, eval function will execute the input which can results code execution |
| **Suggested Remediation** |
| Validated the input before using into eval method |

**Proof of Exploit**  
No Attachments  
  
**Finding #8: HTTP VERB TAMPERING**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 6.4 |
| **Vector** |
|  |
| **Occurrences** |
| 29 |
| **Affected Components** |
|  |
| **Details** |
| By manipulating the HTTP verb it was possible to bypass the authorization on this directory. The scanner sent a request with a custom HTTP verb and managed to bypass the authorization. The attacker can also try any of the valid HTTP verbs, such as HEAD, TRACE, TRACK, PUT, DELETE, and many more. It uses a security control that lists HTTP verbs The security control fails to block verbs that are not listed It has GET functionality that is not idempotent or will execute with an arbitrary HTTP verb |
| **Suggested Remediation** |
| Learns which methods are allowed for each URL. Any attempt to use HTTP methods that are not part of the application’s normal usage will be detected and blocked. The second technique detects non-standard HTTP methods and blocks requests using such methods. In cases where the application uses non-standard methods normally, this mechanism can be easily updated with the allowed methods. |

**Proof of Exploit**  
No Attachments  
  
**Finding #9: SENSITIVE INFORMATION IN RESPONSE HEADERS**

|  |
| --- |
| **Risk** |
| Medium |
| **Severity** |
| Medium |
| **CVSS Base Score** |
| 5.0 |
| **Vector** |
|  |
| **Occurrences** |
| 25 |
| **Affected Components** |
|  |
| **Details** |
| Access-Control-Allow-Origin is a response header used by a server to indicate which domains are allowed to read the response. The Server header contains information about the software used by the origin server to handle the request. The X-Powered-By header contains additional information about the application. Many tools can be used to fingerprint a target. |
| **Suggested Remediation** |
| Access-Control-Allow-Origin : http://13.127.99.204:9090/dnata/ and Specifies an origin, only a single origin can be specified. Remove the Server header from the response. Remove the X-Powered-By header from the response. |

**Proof of Exploit**  
No Attachments