Smart Internz

WEB APPLICATION PENETRATION TESTING: ASSESSING THE SECURITY OF THE MUTILIDAE AND MERCADOLIBRE WEB APPLICATIONS

Submitted by:

Team 2.5

Niladri Mitra (20BIT0381)

Kartik CH (20BIT0340)

Adit Wani (20BIT0188)

Abhirup Konwar (20BIT0181)

1. INTRODUCTION

1.1 Overview

Web application penetration testing is crucial for guaranteeing the security and dependability of modern web systems. This abstract summarizes the importance, goals, methodology, and common vulnerabilities examined during web application penetration testing.

By simulating actual attacks like cross-site scripting (XSS), SQL injection, and authentication bypass, testers can identify issues that could permit unauthorized access, data breaches, or the compromise of critical data. Web application penetration testing also examines vulnerabilities such as cross-site request forgery (CSRF), code injection, unsafe direct object references, insufficient access constraints, and security setup mistakes.

Qualified professionals must perform authorized web application penetration testing following ethical and legal standards. Unauthorized testing may harm systems and have unintended consequences. Organizations can regularly conduct rigorous testing to secure web apps and protect critical data proactively.

1.2 Purpose

The Mutillidae and MercadoLibre websites will undergo web application and penetration testing as part of an extensive assessment of their security architecture and procedures. In this exhaustive testing procedure, the website's source code, network architecture, and user interfaces are just a few of the many meticulously examined components. The testing tries to find and assess any vulnerabilities that may otherwise go unnoticed, leaving the websites vulnerable to exploitation by unwanted actors. It does this by using industry-standard techniques and methodologies.

Through this in-depth research, we aim to identify any loopholes or errors in the conception and application of the security procedures of the websites. The testing procedure tries to imitate the strategies and tactics used by possible adversaries by simulating real-world attack situations, providing insightful information into the platform's overall security resilience.

2. LITERATURE SURVEY

2.1 Existing Problem

The Mutillidae and MercadoLibre websites will now undergo web application and penetration testing because cybersecurity threats offer significant dangers in the current digital environment. Organizations must proactively find and fix vulnerabilities in their online applications due to the continuously rising sophistication of malicious actors and the constant introduction of new attack vectors. Possible issues, including insufficient input validation, a lack of mechanisms for authentication and authorization, shoddy encryption algorithms, and unsafe network setups, can be found and fixed through this testing. The initiative also attempts to solve difficulties with user data privacy, guaranteeing that private data is sufficiently protected against unauthorized access and potential data breaches. The project aims to strengthen the overall security posture of the Mutillidae and MercadoLibre websites by detecting and fixing these issues, increasing user confidence and trust in them.

2.2 Tools used

1. **Nmap Scan:** For finding hosts and services on a computer network, Nmap is a potent network scanning tool. It uses various scanning techniques to obtain information on open ports, active services, and operating systems. Nmap can be used for vulnerability analysis as well as reconnaissance.

2. **WhatWeb**: It is a web vulnerability scanner that determines websites' frameworks and technologies. It analyses HTTP responses and HTML content to identify the presence of particular software components, including content management systems, web servers, programming languages, and more. WhatWeb offers assistance in locating potential security flaws related to particular technologies.

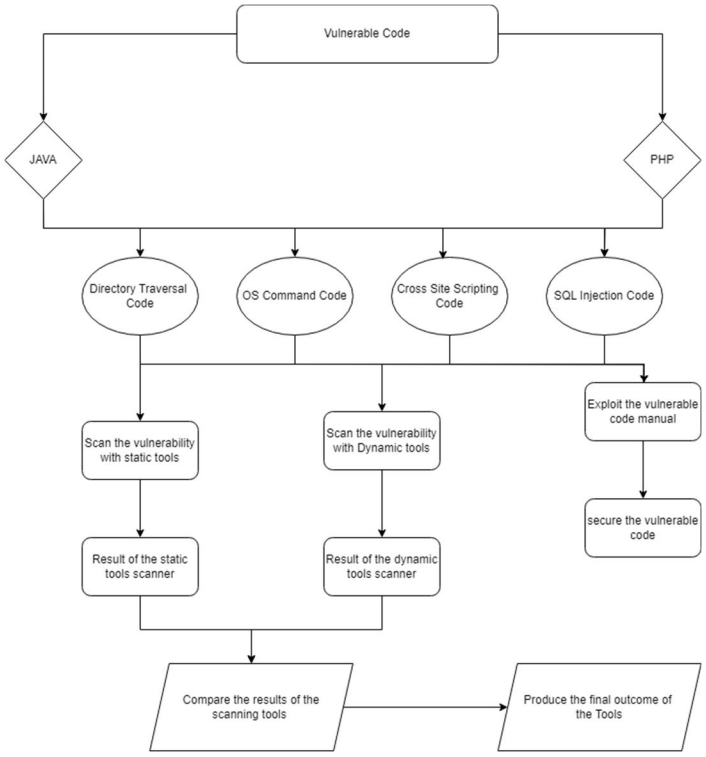
3. **DirBuster:** A directory and file enumeration tool for web applications. It runs a brute-force scan on a web server by repeatedly checking for directories and files. DirBuster assists in finding sensitive files, directories, or information that may be exposed and open to unauthorized access.

4. **SQLmap:** Known as an open-source penetration testing tool, SQLmap takes advantage of web applications' SQL injection flaws. In order to provide a thorough evaluation of the application's security posture concerning database interactions, it automates the process of identifying and exploiting SQL injection problems in database back-ends.

5. **Burp Suite:** Burp Suite is a popular tool for assessing the security of web applications. It includes several features like web proxy tools, manual penetration testing, and vulnerability detection. Cross-site scripting (XSS), SQL injection, and other web application vulnerabilities can be found and exploited using Burp Suite. Penetration testers frequently use it to evaluate and improve the security of web applications.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing:

A. Hardware Requirements:

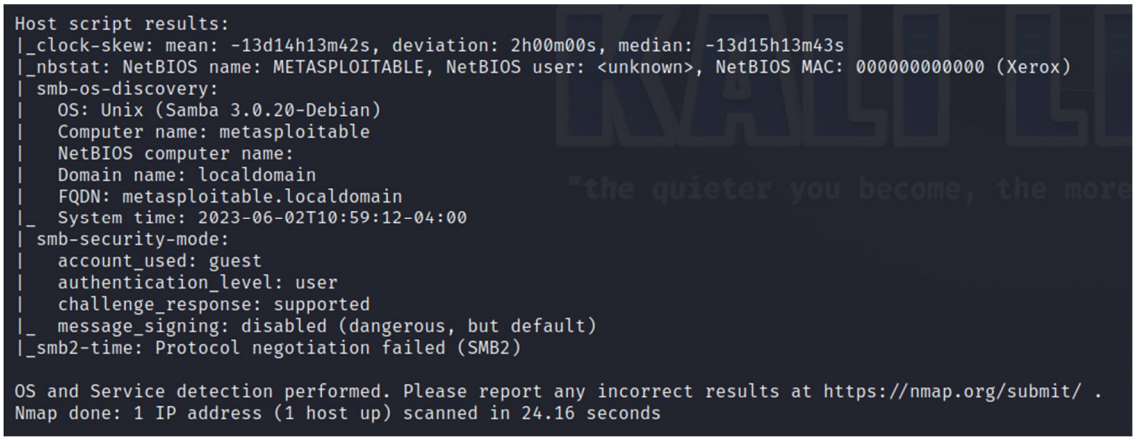
i) Computer with at least 8GB RAM and i5 Processor
ii) Windows 10/Linux Operating System
iii) Wireless Adapters
iv) Network Interface Card (NIC)

B. Software Requirements:

i) VMware
ii) Nmap SCAN
iii) Dirbuster
iv) WhatWeb
v) SQLmap
vi) Burp Suite

4. EXPERIMENTAL INVESTIGATIONS

I) On running Nmap SCAN on mutillidae, the following details about the default scripts, service version and OS were discovered.



II) The following information was retrieved by running WhatWeb on mutillidae.

III) SQLMap displayed the underlying databases, tables and contents of the table for the application mutillidae.

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.50.131/mutillidae/index.php?page=view-someones-blog.php" --data
n=View+Blog+Entries" --cookie="PHPSESSID=c1b8147099d952176e66600b4f2bca1f" --dbs  --batch
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.2#stable}
|_ -| . [(]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is i
all applicable local, state and federal laws. Developers assume no liability and are not resp
ram

[*] starting @ 12:47:24 /2023-06-16/
```

```
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

```
Database: owasp10
[6 tables]
+-----------------+
| accounts        |
| blogs_table     |
| captured_data   |
| credit_cards    |
| hitlog          |
| pen_test_tools  |
+-----------------+
```

```
Database: owasp10
Table: credit_cards
[5 entries]
+------+-----+-------------------+------------+
| ccid | ccv | ccnumber          | expiration |
+------+-----+-------------------+------------+
| 1    | 745 | 4444111122223333  | 2012-03-01 |
| 2    | 722 | 7746536337776330  | 2015-04-01 |
| 3    | 461 | 8242325748474749  | 2016-03-01 |
| 4    | 230 | 7725653200487633  | 2017-06-01 |
| 5    | 627 | 1234567812345678  | 2018-11-01 |
+------+-----+-------------------+------------+
```

III) Dirbuster showed the following directories were being used in mutillidae.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://192.168.50.131:80/

Scan Information | Results - List View: Dirs: 28 Files: 18 | Results - Tree View | Errors: 0 |

Type	Found	Response	Size
Dir	/twiki/bin/	403	473
Dir	/twiki/	200	1039
Dir	/phpMyAdmin/main/	200	670
Dir	/phpMyAdmin/index/	200	676
Dir	/phpMyAdmin/	200	643
Dir	/mutillidae/register/	200	2000
Dir	/mutillidae/login/	200	183
Dir	/mutillidae/index/	200	326
Dir	/mutillidae/images/	200	6176
Dir	/mutillidae/home/	200	183
Dir	/mutillidae/	200	326
Dir	/index/	200	1096
Dir	/icons/	200	160
Dir	/dvwa/security/	302	335
Dir	/dvwa/login/	200	297
Dir	/dvwa/index/	302	335
Dir	/dvwa/dvwa/images/	200	2221
Dir	/dvwa/dvwa/	200	1593
Dir	/dvwa/docs/	200	1089
Dir	/dvwa/about/	302	335
Dir	/dvwa/	302	335
Dir	/dav/x6NSOFWf.htm/	200	891
Dir	/dav/	200	1622
Dir	/cgi-bin/	403	471
Dir	/	200	1094

Current speed: 45 requests/sec (Select and right click for more options)

Average speed: (T) 44, (C) 38 requests/sec

IV) The following open ports were discovered in the mutillidae application.

[1] Port 512 (rlogin - Remote Program Execution):

Exploits:

a. Username/Password Sniffing: As rlogin transmits data in clear text, an attacker positioned on the network could intercept and capture usernames and passwords. This can be achieved using packet sniffing tools or techniques like ARP spoofing.
b. Brute-Force Attacks: Attackers can launch brute-force attacks against the rlogin service on port 512 to systematically guess usernames and passwords. By using automated tools or scripts, they can attempt to log in with various combinations until they find a valid credential.

Mitigation strategies:

a. Use Secure Alternatives: Instead of rlogin, use more secure remote access protocols like Secure Shell (SSH). SSH provides encrypted communication, strong authentication mechanisms, and improved security features compared to rlogin. Ensure that SSH is properly configured and up to date.
b. Encrypt Network Traffic: If rlogin is still required, consider using a secure tunneling mechanism like Virtual Private Network (VPN) to encrypt the network traffic between client and server. This adds an additional layer of protection and helps prevent unauthorized interception and sniffing of sensitive information.

[2] Port 514 (Syslog Protocol):

Exploits:

a. Log Injection: Attackers may attempt to inject malicious or misleading log messages into the syslog server listening on port 514. By crafting log messages with specially crafted content, they can attempt to manipulate the system or deceive administrators, potentially leading to misinterpretation of events or false alarms.
b. Log Tampering: If an attacker gains unauthorized access to the syslog server or intercepts log messages being sent to port 514, they can tamper with the log data. This can involve modifying or deleting log entries, disguising or removing evidence of their activities, or altering timestamps, making it harder to detect and investigate security incidents.

Mitigation strategies:

a. Secure Access to Syslog Servers: Restrict access to syslog servers listening on port 514 by allowing connections only from trusted sources or specific IP addresses. Implement strong authentication mechanisms, such as username/password or certificate-based authentication, to prevent unauthorized access.
b. Filter and Validate Log Messages: Apply input validation and filtering mechanisms to log messages received on port 514. This helps prevent log injection attacks by ensuring that only valid log data is accepted and processed. Employ techniques such as whitelisting, blacklisting, and regular expression matching to filter out suspicious or malicious log entries.

[3] Port 1099 (Java Remote Method Invocation):

Exploits:

a. RMI Registry Enumeration: Attackers can perform RMI Registry enumeration to identify and list available remote objects and their associated methods. This information can be used to gain insights into the application's structure, potentially revealing sensitive details or providing a roadmap for further exploitation.
b. Remote Code Execution (RCE): If the RMI Registry or the remote objects it manages are vulnerable to remote code execution vulnerabilities, an attacker can craft malicious RMI requests to execute arbitrary code on the target system. This can lead to unauthorized access, data exfiltration, or compromise of the entire system.

Mitigation strategies:

a. Secure Network Access: Restrict network access to the RMI Registry service on port 1099. Allow connections only from trusted sources or specific IP addresses to minimize the attack surface.
b. Access Controls and Authentication: Implement access controls and strong authentication mechanisms to restrict access to the RMI Registry and the remote objects it manages. Enforce proper authorization checks and use secure authentication protocols to prevent unauthorized access.

[4] **Port 2049 (NFS - Network File System):**

Exploits:
a. Unauthorized access: Exploiting weak or misconfigured NFS permissions to gain unauthorized access to shared files.
b. Man-in-the-middle attacks: Intercepting NFS traffic to eavesdrop on or modify the communication between client and server.

Mitigation strategies:
a. Implement proper access controls and restrict NFS access to trusted hosts only.
b. Use NFS version 4 with secure configurations, such as Kerberos authentication and Transport Layer Security (TLS) encryption.

[5] **Port 3306 (MySQL - Database Management System):**

Exploits:
a. Brute force attacks: Repeatedly trying different username and password combinations to gain unauthorized access to MySQL databases.
b. SQL injection attacks: Exploiting vulnerabilities in web applications to execute malicious SQL queries against the MySQL database.

Mitigation strategies:
a. Enforce strong passwords and implement account lockout policies to protect against brute force attacks.
b. Sanitize and validate user inputs to prevent SQL injection vulnerabilities in web applications.

[6] **Port 5432 (PostgreSQL - Database Management System):**

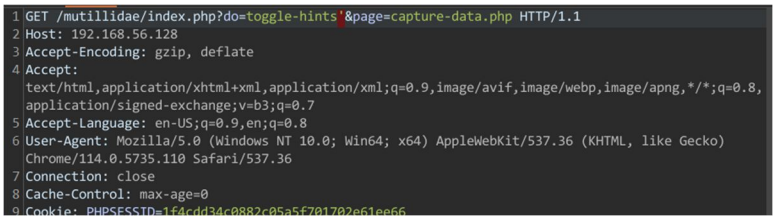
Exploits:
a. Default or weak credentials: Exploiting default or easily guessable usernames and passwords to gain unauthorized access to PostgreSQL databases.
b. SQL injection attacks: Leveraging vulnerabilities in web applications to execute malicious SQL queries against the PostgreSQL database.

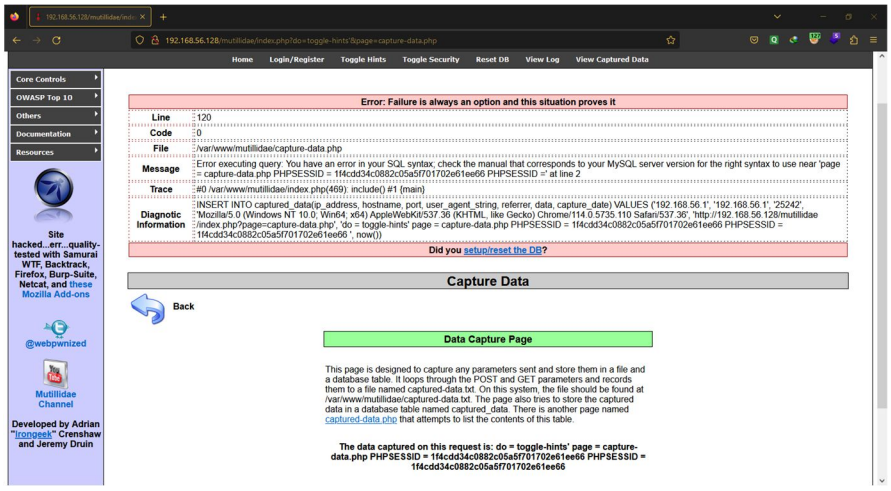
Mitigation strategies:
a. Change default credentials and use strong passwords for PostgreSQL database accounts.
b. Implement network segmentation and rate limiting to protect against DoS attacks on the PostgreSQL server.

IV) SQL Injection Vulnerabilities in mutillidae application:-

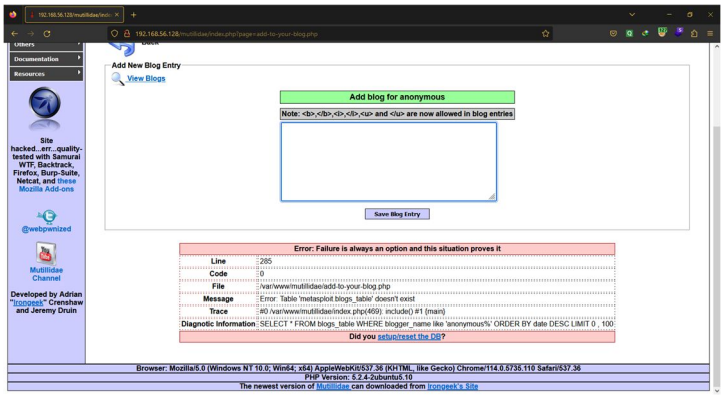
SQL Injection	
Risk Classification	CWE-89, CWE-94, CWE-116
Description	The parameters – **blog, entry, password**, and **username** appear to be vulnerable to SQL injection attacks. A single quote was submitted in the username parameter, and a database error message was returned. Two single quotes were then submitted and the error message disappeared.
Impact	i) Sensitive information such as the user information, OS, network configuration etc can be disclosed to the attacker. ii) The attacker can gain access to the server and can even take control of the entire system. iii) DoS attacks can be launched on the server using resource-intensive and infinite loops.
Recommendation	It is recommended to implement the following configurations: 1. **Input Validation:** Comparing against a whitelist of permitted values, allowing only alphanumeric characters in the field and not any other syntax. 2. **Least Privilege:** The users of the application should get the least privilege possible so that the command cannot cause any substantial damage. 3. **Error Messages:** It's recommended to not display SQL Error messages in the responses.
Affected URL and Ports	http://192.168.56.128/mutillidae/index.php

On entering ‘ in the request header, the session id along with OS details is displayed.





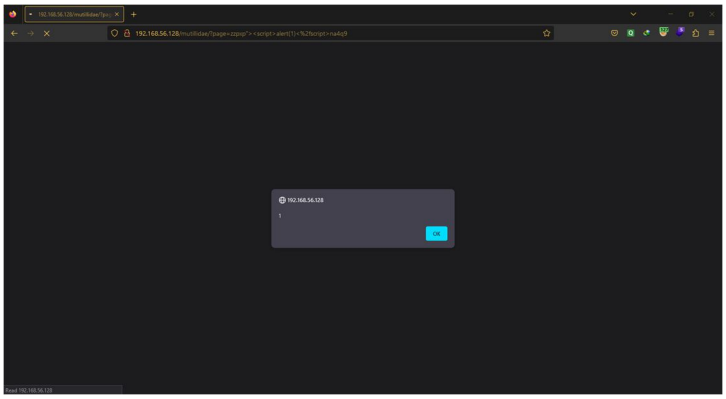
On entering the payload 32945237' or 4475=4475— in the request parameter in URL encoded format, an SQL error message was received, showing that it is vulnerable.



IV) XSS Vulnerabilities in mutillidae application:-

Cross-Site Scripting (reflected)	
Risk Classification	CWE-79, CWE-80, CWE-116, CWE-159
Description	• Reflected cross-site scripting vulnerabilities occur when data is echoed into an application's response without proper validation. • Attackers can exploit this vulnerability to execute their JavaScript code within a user's browser session. • Users can be tricked into executing the attacker's request through various means, such as malicious URLs in emails or instant messages.
Impact	i) Sensitive information such as the user information, OS, network configuration etc can be disclosed to the attacker. ii) Attacker may be able to retrieve items that are normally protected from direct access, such as application configuration files, the source code for server-executable scripts, or files with extensions that the web server is not configured to serve directly.
Recommendation	• Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized. • User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).
Affected URL and Ports	1. **http://192.168.56.128/mutillidae/index.php** 2. **http://192.168.56.128/mutillidae/**

On entering zzpxp"><script>alert(1)</script>na4q9 in the request parameter, a pop up is generated.

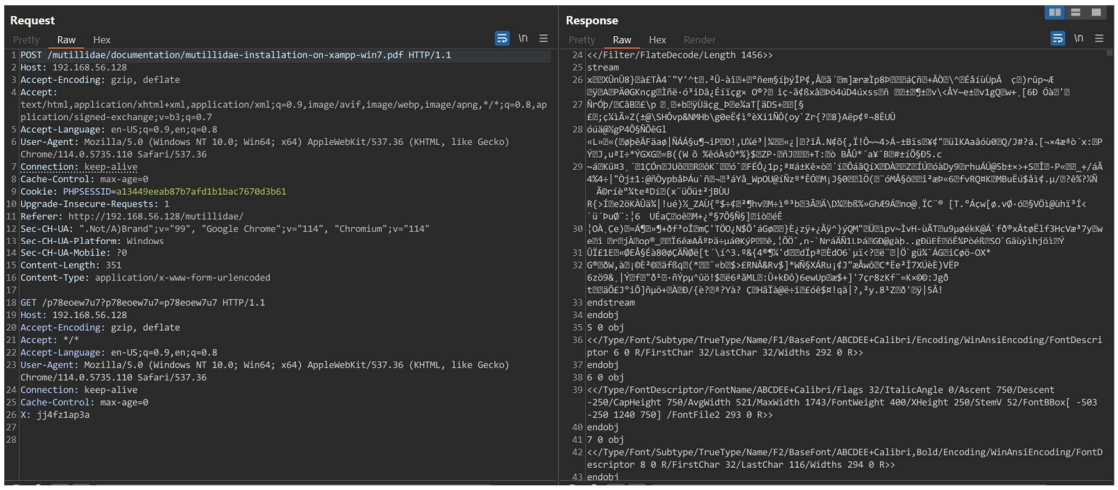


The same pop-up is visible on entering the code in the password, username, blog_entry and page parameters of the index.php page.

V) Client Side Desync in mutillidae application:-

Client-Side Desync	
Risk Classification	CWE-444
Description	The server appears to be vulnerable to client-side desync attacks. The server ignored the Content-Length header and did not close the connection, leading to the smuggled request being interpreted as the next request.
Impact	The web server fails to correctly process the Content-Length of POST requests. Exploiting this behavior, an attacker can force a victim's browser to desynchronize its connection with the website, typically leading to XSS.
Recommendation	• You can resolve this vulnerability by patching the server so that it either processes POST requests correctly, or closes the connection after handling them. • You could also disable connection reuse entirely, but this may reduce performance. You can also resolve this issue by enabling HTTP/2.
Affected URL and Ports	**http://192.168.56.128/mutillidae/documentation/mutillidae-installation-on-xampp-win7.pdf**

A POST request was sent to the path '/mutillidae/documentation/mutillidae-installation-on-xampp-win7.pdf' with a second request sent as the body. The second request sent as body was interpreted as the next request, since the previous connection was not closed.



VI) File Traversal in mutillidae application:-

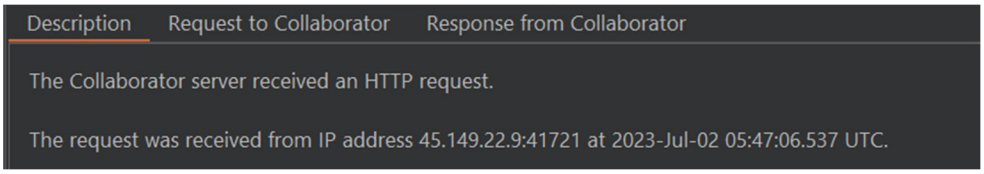
File Traversal	
Risk Classification	CWE-22, CWE-23, CWE-35, CWE-36
Description	The page parameter is vulnerable to path traversal attacks, enabling read access to arbitrary files on the server.
Impact	i) Sensitive information such as the user information, OS, network configuration etc can be disclosed to the attacker. ii) Attacker may also gain access to files containing configuration data, passwords, database records, log data, source code, and program scripts and binaries.
Recommendation	It is recommended to implement the following configurations: 1. After validating user input, the application can use a suitable file system API to verify that the file to be accessed is actually located within the base directory used by the application. 2. User-controllable data should be strictly validated before being passed to any file system operation. In particular, input containing dot-dot sequences should be blocked. 3. The directory used to store files that are accessed using user-controllable data can be located on a separate logical volume to other sensitive application and operating system files, so that these cannot be reached via path traversal attacks. a. For Unix – a chrooted file system b. For Windows - mounting the base directory as a new logical drive and using the associated drive letter to access its contents.
Affected URL and Ports	**http://192.168.56.128/mutillidae/** **http://192.168.56.128/mutillidae/index.php**

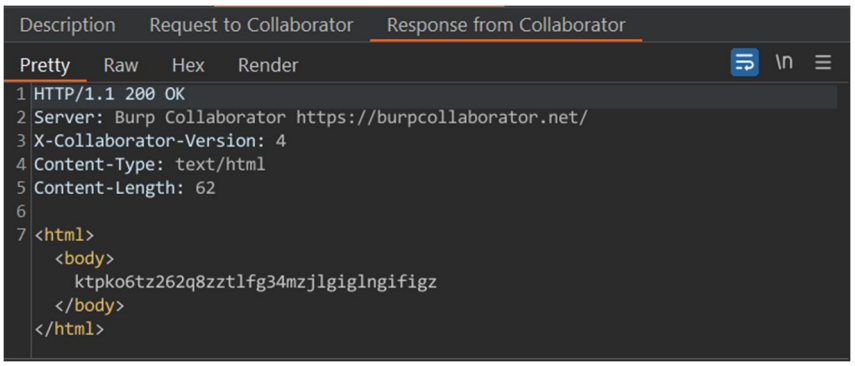
The payload ../../../../../../../../../../../../../../../../**etc/passwd** was submitted in the page parameter. The requested file was returned in the application's response.

VII) External Service Interaction in mercadolibre application:-

External Service Interaction	
Risk Classification	CWE-918
Description	It is possible to induce the application to perform server-side HTTP requests to arbitrary domains. The application performed an HTTP request to the specified domain. The behavior appears to be asynchronous, and the Collaborator interaction occurred approximately 98 seconds after the scan of the item was completed.
Impact	The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself.
Recommendation	It is recommended to implement the following configurations: 1. Blocking network access from the application server to other internal systems. 2. Remove any services available on the local loopback adapter. 3. Implement a whitelist of permitted services and hosts, and block any interactions that do not appear on this whitelist.
Affected URL and Ports	**https://mercadolibre.com**

The payload **gwg8x5p132acomm2trukh8skmbs4gu4nzbpyfm4.oastify.com** was submitted in the HTTP Referer header and the application performed an HTTP request to the specified domain.

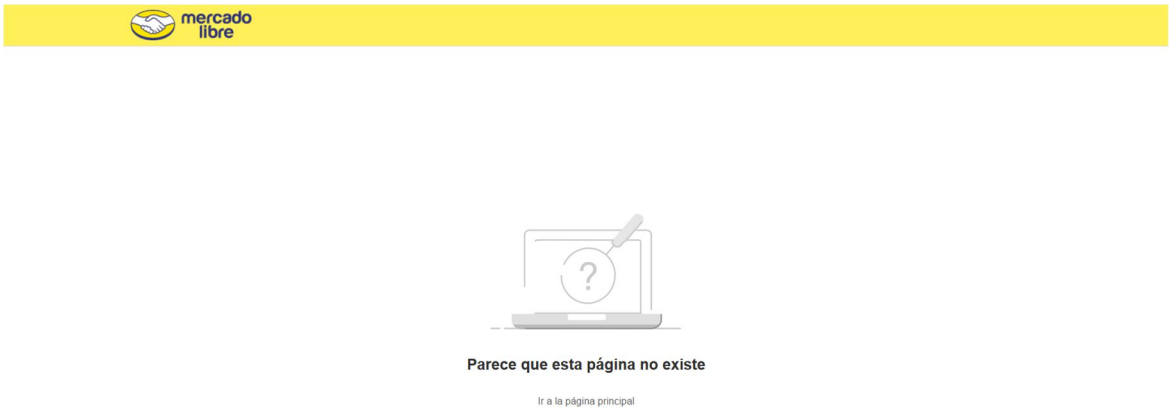




VIII) Clickjacking in mercadolibre application:-

Clickjacking	
Risk Classification	CWE-693
Description	By inducing victim users to perform actions such as mouse clicks and keystrokes, the attacker can cause them to unwittingly carry out actions within the application that is being targeted. This technique allows the attacker to circumvent defenses against cross-site request forgery, and may result in unauthorized actions.
Impact	The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself.
Recommendation	To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.
Affected URL and Ports	**https://mercadolibre.com**

On using a different web page in the application using iframe, it gets imposed on top of the real application, thus making it vulnerable to clickjacking attacks.

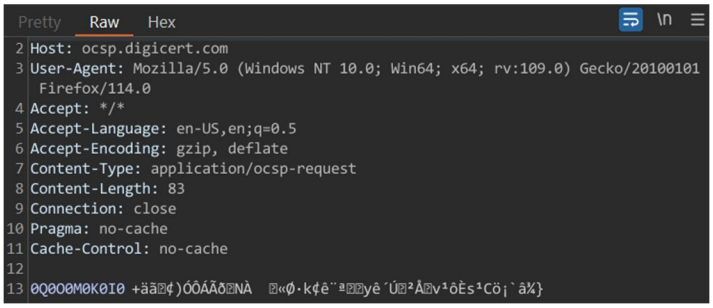


IX) Incorrect Content Type in mercadolibre application:-

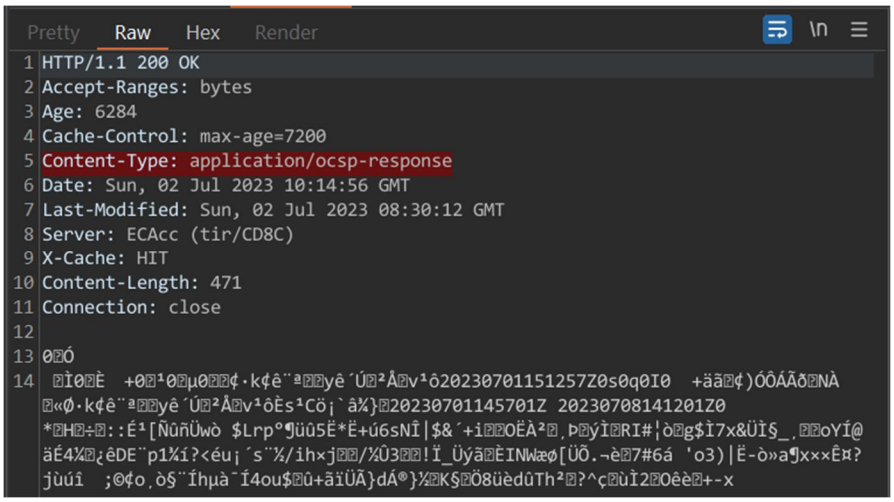
Incorrect Content Type	
Risk Classification	CWE-16, CWE-436
Description	The response states that the content type is application/ocsp-response. However, it actually appears to contain unrecognized content. If the URL path can be manipulated to end with ".html", browsers may interpret the response as HTML.
Impact	The ability to send requests to other systems can allow the vulnerable server to be used as an attack proxy. By submitting suitable payloads, an attacker can cause the application server to attack other systems that it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself.
Recommendation	To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.
Affected URL and Ports	**https://mercadolibre.com**

On sending the request containing unrecognized content, a response is returned with unrecognized characters which the browser interprets as application response.

Request:



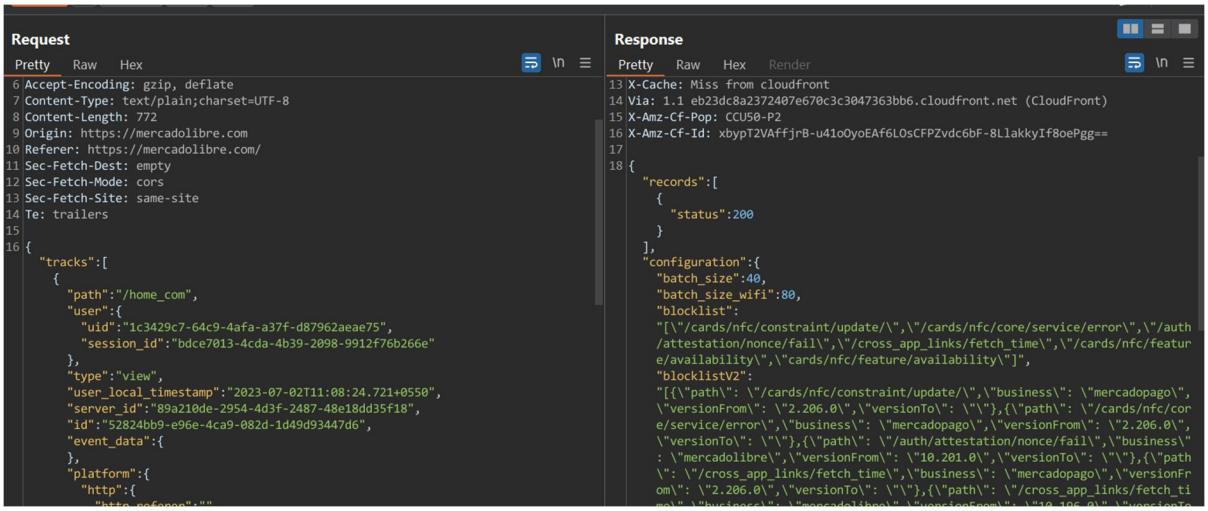
Response:



X) Cacheable HTTPS response in mercadolibre application:-

Cacheable HTTP Response	
Risk Classification	CWE-524, CWE-525
Description	Unless directed otherwise, browsers may store a local cached copy of content received from web servers. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.
Impact	i) Information Disclosure: If a response containing sensitive information, such as user credentials, personal data, or confidential documents, is cached, it can be accessed by unauthorized individuals. This could occur if the caching mechanism is not configured correctly or if the cache is not properly invalidated when sensitive data is involved. ii) Session Management Issues: Caching responses can interfere with proper session management. For example, if a response containing session identifiers or tokens is cached, it could result in session fixation or session hijacking attacks. Attackers could potentially obtain valid session information and impersonate users. iii) CSRF (Cross-Site Request Forgery) Exploitation: Caching can make CSRF attacks more challenging to mitigate. If a CSRF token is cached, an attacker might be able to reuse it to execute unauthorized actions on behalf of a victim, even after the user has logged out or the token has expired.
Recommendation	Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content: • Cache-control: no-store • Pragma: no-cache
Affected URL and Ports	**https://api.mercadolibre.com**

The following sensitive information was cached and stored in the local machine which was retrieved by the appropriate request as used below.



5. ADVANTAGES AND DISADVANATGES

5.1 Advantages

1) Finding Vulnerabilities: Penetration testing aids in identifying weaknesses in the applications. Essential insights into the flaws can be obtained that nefarious actors can attack by finding and documenting vulnerabilities like the ones above.
2) Risk Prioritization: Penetration testing gives clear insight into the dangers connected to the found vulnerabilities. This enables the user to order remediation actions by each vulnerability's seriousness and potential consequences.
3) Security Control Validation: Penetration testing aids in confirming the efficiency of the current security policies. It can determine whether the security measures, such as input validation, access controls, or session management, are working as intended by attempting to exploit vulnerabilities.

5.2 Disadvantages

1) Limited Scope: Within a predetermined scope, penetration testing often concentrates on a particular group of targets. It might only cover some facets of the infrastructure as a whole or find vulnerabilities in parts outside its purview.
2) Service Interruptions: There is a chance that penetration testing will have unforeseen consequences, such as service interruptions or unanticipated side effects on the systems being examined. The testing process must be carefully planned and coordinated to minimise any influence on production environments.

6. FUTURE SCOPE

The following is the future range of web application penetration testing:

1) New Web Technologies: Specialized penetration testing techniques and tools will be required to assess the security of new web technologies like serverless architectures, microservices, and single-page applications (SPAs). Web application security testers must stay up to date and adapt as needed.
2) Internet of Things (IoT): The scope of web application pen-testing will be widened to include security assessments of these IoT apps as IoT devices grow and more are outfitted with web interfaces or APIs. IoT devices must have their web interfaces, APIs, communication protocols, and overall security evaluated in order to accomplish this.
3) Mobile application integration: Many web applications now include mobile web interfaces or native mobile apps for their mobile counterparts. Future web application pen-testing will likely include an analysis of the security of these integrated mobile components, guaranteeing the safety of sensitive data and preventing vulnerabilities particular to mobile platforms.

7. CONCLUSION

The Mutillidae and Mercadolibre penetration testing project found SQL injection, XSS, client-side desync, file traversal, external service interaction, clickjacking, incorrect content type, and cacheable HTTP response issues. This project revealed application and system shortcomings, helping prioritize risk mitigation.

8. BIBLIOGRAPHY

1. https://portswigger.net/web-security/cross-site-scripting/reflected
2. https://portswigger.net/web-security/sql-injection/cheat-sheet
3. https://portswigger.net/web-security/file-path-traversal
4. https://portswigger.net/research/browser-powered-desync-attacks
5. https://portswigger.net/research/cracking-the-lens-targeting-https-hidden-attack-surface
6. https://portswigger.net/web-security/clickjacking
7. https://portswigger.net/web-security/information-disclosure