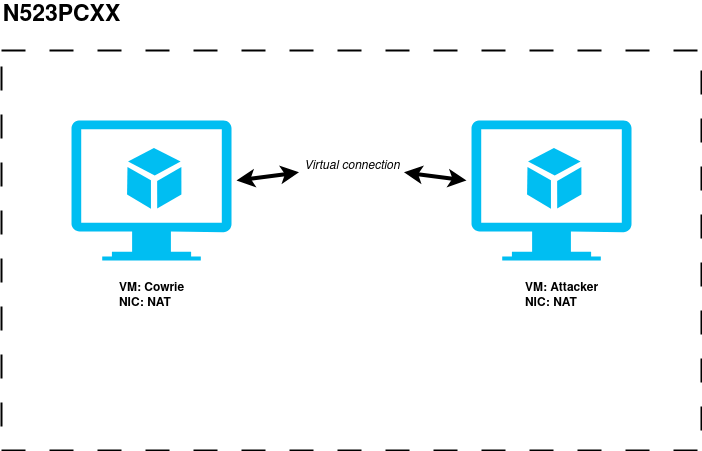
# TOPOLOGY SETUP



Notes:

* You will use two VMs throughout this lab: **Cowrie** and **Attacker**
* When configuring the Cowrie VM you will use two accounts: **SOC User** **(analyst)** and **Cowrie User**
* Both VMs should be set to NAT so they are automatically allocated an IP address from VMWare

# COWRIE HONEYPOT VM SETUP

## Setup VM and import into VMWare Workstation

* Make a copy of our Ubuntu\_24\_04 VM and name one **Cowrie** and one **Attacker**

* Import the **Cowrie** VM into VMWare Workstation

* Change the allocated RAM to 4GB

* Rename the VM (optional)

* Ensure the network adapter is set to NAT

* Take a snapshot of the VM, name it “Base\_cowrie” (RightClick > SnapShot > SnapShot Manager > Take SnapShot)

* Power on the **Cowrie** VM and when asked select “*I copied it*”

* Login with password: cyberops

* Open a Terminal window and make a note of the IP address using command

***ip address***

## Install prerequiste packages [SOC User]

• Install cowrie using the following commands. Note: install commands can be combined using && but some are shown here separately for clarity.   
  
  
**Required packages:**

|  |
| --- |
| **sudo apt update && sudo apt upgrade -y**  **sudo apt install -y git python3-virtualenv libssl-dev libffi-dev build-essential**  **sudo apt install –y libpython3-dev python3-minimal authbind virtualenv**  **sudo apt install –y openssh-server** |

**Creation of the Cowrie user:**

|  |
| --- |
| # this users (cowrie) will run as a non-root user  **sudo adduser --disabled-password cowrie**  # in this terminal we will be logged in as the cowrie user  **sudo su - cowrie** |

## Install Cowrie [SOC User]

* Open a second terminal window to download and install cowrie from our SOC User account (leav the other terminal window open

**Downloading and configuration:**

|  |
| --- |
| # download the cowrie installer from github  wget https://github.com/cowrie/cowrie/archive/refs/tags/v2.4.0.tar.gz  # list directory to check cowrie downloaded  # extract and rename the directory  # move the cowrie directory to the cowrie user home folder  # take ownership of the cowrie dirtectory  ls  tar -xvzf v2.4.0.tar.gz  mv cowrie-2.4.0 cowrie  sudo mv cowrie /home/cowrie  sudo chown -R cowrie:cowrie /home/cowrie |

## Setup virtual environment for cowrie to run in [Cowrie User**]**

* Return to the original terminal window (Cowrie user) to continue using cowrie

**Create python virtual environment:**

|  |
| --- |
| cd cowrie  virtualenv --python=python3 cowrie-env  source cowrie-env/bin/activate  pip install --upgrade pip  pip install --upgrade -r requirements.txt  deactivate |

**Configure Cowrie [Cowrie User]**

* Configure with minimum settings

**Minimal configuration of Cowrie:**

|  |
| --- |
| *cd /home/cowrie/cowrie/etc*  *# make a copy of cowrie.cfg.dist and name cowrie.cfg*  *cp cowrie.cfg.dist cowrie.cfg*  # edit cowrie.cfg to setup cowrie HoneyPot (do not use sudo)  *nano cowrie.cfg*  # change name from svr04  honeypot]  hostname = cowrieInstallation  # change ssh port from 2222 to 22  [ssh]  listen\_endpoints = tcp:22:interface=0.0.0.0  # change telnet port from 2023 to 23  [telnet]  enabled = true  listen\_endpoints = tcp:23:interface=0.0.0.0  # save the file and exit  CTRL + X, Y, Enter |

## Install authbind [SOC User]

* Install authbind and create necessary files.
* On Linux, ports below are “privileged” meaning normally only processes started by root can bind to them. The below commands use authbind so Cowrie’s non-root process can still use those ports safely.  
    
  **Configuration of *authbind*:**  
    
  Here we are editing the port 22 and 23 at the same time:

|  |
| --- |
| sudo apt install authbind  # create an empty file named 22 and 23 in */etc/authbind/byport/*  # The filename corresponds to the port number and is how authbind knows which ports a user is allowed to bind to.  sudo touch /etc/authbind/byport/22  # change ownership of that file to the cowrie user and group. This means only the Cowrie user will have permission to use port 22 through authbind.  sudo chown cowrie:cowrie /etc/authbind/byport/22  # set read/write/execute permissions for owner and group (cowrie), and no permissions for others. The execute bit here tells authbind that binding to the port is allowed.  sudo chmod 770 /etc/authbind/byport/22  sudo touch /etc/authbind/byport/23 sudo chown cowrie:cowrie /etc/authbind/byport/23 sudo chmod 770 /etc/authbind/byport/23 |

**Install and configure SSH [SOC User]**

* Install and configure SSH and Telnet

|  |
| --- |
| sudo apt-get install ssh  sudo nano /etc/ssh/sshd\_config  edit: Port 3393 ListenAddress 0.0.0.0  # save the file and exit  CTRL + X, Y, Enter  # restart ssh service  sudo systemctl restart ssh |

**Basic usage of Cowrie [Cowrie User]**

* Start/Stop/Status of cowrie

**Start and verification:**

|  |
| --- |
| # if not still logged into cowrie and in the virtual environment issue below commands. If still logged in skip these commands and start cowrie  sudo su – cowrie cd /home/cowrie/cowrie  source cowrie-env/bin/activate  # start cowrie and check status  bin/cowrie start bin/cowrie status |

# ATTACKER VM SETUP

**Setup VM and import into VMWare Workstation**

* Import the **Attacker** VM into VMWare Workstation

* Change the allocated RAM to 4GB

* Rename the VM (Optional)

* Ensure the network adapter is set to NAT

* Take a snapshot of the VM, name it “Base\_Attacker” (RightClick > SnapShot > SnapShot Manager > Take SnapShot)

* Power on each VM and when asked select “*I copied it*”

* Login with password: cyberops

* Open a Terminal window and make a note of the IP address using command

***ip address***

## Install Cowrie Detect and Tools

## Installation of Cowrie Detect

|  |
| --- |
| **Installing dependencies** sudo apt-get update && sudo apt-get upgrade  sudo apt install -y git python3-venv nmap   **Downloading of the script Cowrie Detect** git clone <https://github.com/boscutti939/Cowrie_Detect.git>   **Creation and activation of Python environment** python3 -m venv cowrie-env  source cowrie-env/bin/activate  pip install python3-nmap paramiko  deactivate |

## Set up of the Detection

|  |
| --- |
| **#** To have a clean return modify the file “cowrie\_detect” add the letter “r” before the quotation mark: pattern = re.compile(r"[0-9A-Fa-f]{2}-[0-9A-Fa-f]{2}-[0-9A-Fa-f]{2}")  Then go on testing part to see if the Cowrie is correctly setup. |

## 

# TESTING

* Using both the **Cowrie** and **Attacker** VMs we will now test the honeypot.
* With both VMs up and running make a note of each IP address
* Test connectivity between the VMs by issuing a ping request for each VM

Cowrie VM IP : 192.168.x.x

Attacker VM IP: 192.168.x.x

## Start Cowrie Honeypot Running [Cowrie User] [Cowrie VM]

* On the **Cowrie** VM start the HoneyPot running and check the status

|  |
| --- |
| # Launch Cowrie on the Cowrie VM sudo su - cowrie  cd /home/cowrie/cowrie  source cowrie-env/bin/activate  bin/cowrie start  bin/cowrie status |

## Run cowrie\_detect script [Soc User] [Attacker VM]

* On the **Attacker** VM run the cowrie\_detect script to evaluate if the remote host is a honeypot, where 192.168.x.x is the IP address of your HoneyPot

|  |
| --- |
| source cowrie-env/bin/activate  cd Cowrie\_Detect/  python cowrie\_detect.py 192.168.x.x   Kind of return: Total Score: 105.0 / 110 (95.45%) **Verdict: A Cowrie honeypot with slightly changed values.** |

**Q Was the remote device detected as a HoneyPot?**

Yes. Since no changes were made yet to Cowrie’s default configuration, the detection score remained high (around 95–100%), clearly identifying the system as a Cowrie honeypot.

**Q What variables did the script check for?**

**You can find the variables which are check on the following tab:**

|  |  |
| --- | --- |
| Check | What it looked at |
| OS match in /proc/version | uname result or version string |
| CPU info | /proc/cpuinfo |
| Memory info | /proc/meminfo |
| Mount points | /proc/mounts |
| Users X | Present in: /etc/passwd, /etc/shadow, /etc/group |
| Hostnames | From /etc/hosts, /etc/hostname |
| Login banner | /etc/issue |
| OUI/MAC vendor test | Failure (due to no internet connexion) |
| Nmap | Not installed (so fil of the scan) |

## Change default Cowrie Configuration [Cowrie User] [Cowrie VM]

* On the **Cowrie** VM stop the HoneyPot running

|  |
| --- |
| sudo su – cowrie  cd /home/cowrie/cowrie  bin/cowrie stop  bin/cowrie status |

## Saving (take a Snapshot)

**Call your Snapshot:** pre-customisations

**Understanding the files to be modified**

* Customisations to the default cowrie configuration can be made to make the honeypot detection by an attacker more difficult. Default values can be modified, new directories added, and system configurations changed as appropriate. This is to ensure that the honeypot system is not easily detected as a honeypot by automated tools such as **cowrie\_detect.py** . Some examples of changes that can be made to default settings are:

|  |  |  |
| --- | --- | --- |
| **Type of Element** | **File or Directory** | **Purpose** |
| Configuration File | /home/cowrie/cowrie/etc/cowrie.cfg | Main configuration file for Cowrie |
| Virtual Filesystem (hidden) | /home/cowrie/cowrie/share/cowrie/fs.pickle | Fake internal file mapping used by Cowrie (not directly visible) |
| Visible Fake Filesystem | /home/cowrie/cowrie/honeyfs/ | Files and folders seen by the attacker inside the fake shell |

The following are various fake filesystems contained within cowrie that can be edited to make the HoneyPot appear more genuine to an attacker:

Configuration File (**./etc/cowrie.cfg**)

Pickle Filesystem (**./share/cowrie/fs.pickle**): This is a fake virtual filesystem on Cowrie honeypot as the files in this directory do not exist. Attackers could list these files in the directory by issuing the Linux “ls” command

Filesystem (**./honeyfs**): This is the directory of Cowrie file system where a tree of files/folders could be placed. The files in this directory would be visible to the attackers and the files would be available for viewing.

*/home/cowrie/cowrie/honeyfs/etc/motd - post-login banner*

*/home/cowrie/cowrie/honeyfs/proc/cpuinfo – CPU information*

*/home/cowrie/cowrie/honeyfs/proc/meminfo – memory information*

*/home/cowrie/cowrie/honeyfs/proc/mounts – the drive mount*

*/home/cowrie/cowrie/honeyfs/proc/version – version of the system /home/cowrie/cowrie/honeyfs/proc/net/arp – address resolution protocol of the server*

* For the purpose of this exercise we will make minor changes and rerun the detect script. Change the CPU information to the details below

|  |
| --- |
| cd /home/cowrie/cowrie/honeyfs/proc/ nano cpuinfo  **replace the line:** Intel(R) Core(TM)2 Duo CPU E8400 @ 3.70GHz  **Same for the memory information:** nano meminfo *\*\* (change some random values)* |

* Start the HoneyPot on **Cowrie** VM

|  |
| --- |
| cd /home/cowrie/cowrie  bin/cowrie start  bin/cowrie status |

* On the **Attacker** VM run the cowrie\_detect script to evaluate if the remote host is a honeypot, where 192.168.5.130 is the IP address of your HoneyPot

|  |
| --- |
| cd Cowrie\_Detect/  python cowrie\_detect.py 192.168.x.x |

**Q Was detection of the remote computer as a HoneyPot any different? (it may take a little time to make sufficient changes**

Like you can see the score will decrease little bit due to the different modification:

Total Score: 100.0 / 110 (90.91%)

Verdict: A Cowrie honeypot with slightly changed values.

**Simulating an Attacker and checking Logs [Soc User] [Attacker VM]**

* On the **Attacker** VM ssh into the HoneyPot (use username:password root:password)and issue some commands, where 192.168.x.x is the IP address of your HoneyPot   
    
  **Connect via SSH from Attacker VM**

|  |
| --- |
| ssh root@x.x.x.x  **Login with:** **Username:** root **Password:** password   **Then execute:** ls  pwd  cd /home  ls  cd /phil  ls  exit |

**View Logs from the Cowrie VM [Cowrie User][Cowrie VM]**

* On the **Cowrie** VM check the log file to see if you can see evidence of the remote connection

|  |
| --- |
| **Cowrie logged during the attack:**  cd /home/cowrie/cowrie/var/log/cowrie  ls  cat cowrie.json   **Examples of return:** "eventid": "cowrie.login.success",  "eventid": "cowrie.command.input",  "input": "cd /phil",  "eventid": "cowrie.session.closed" |

**Q Could you see the commands you issued as the attacker?**

Yes – Cowrie captures all interactions made in the fake shell. Commands like ls, cd /home, and exit are logged in detail in the cowrie.json file.