

# Configuration Manual

MSc Research Project

MSCCYBETOP

Darragh Goslin

Student ID: x20141416

School of Computing

National College of Ireland

Supervisor: Mark Monaghan

**MSc Project Submission Sheet****School of Computing****Student Name:** Darragh Goslin**Student ID:** x20141416@student.ncirl.ie**Programme:** MSc in Cyber Security**Year:** 2024**Module:** MSc Research Project**Supervisor:** Mark Monaghan**Submission Due Date:** Monday 12th August 2024**Project Title:** Using honeypots to develop an understanding of intrusion techniques.**Word Count:** 1891**Page Count:** 34

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Darragh Goslin**Date:** 10/08/2024**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a <b>HARD COPY</b> of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
<b>Signature:</b>	
<b>Date:</b>	
<b>Penalty Applied (if applicable):</b>	

## Contents

1 Introduction.....	3
2 Configuring Cloud Environment .....	3
3 Creating Virtual Machines .....	4
4 Installing Cowrie.....	10
5 Collecting Logs.....	18
6 Processing Logs .....	20

# Configuration Manual

Darragh Goslin

Student ID: x20141416@student.ncirl.ie

## 1 Introduction

This configuration manual outlines the steps involved in implementing a cloud based honeypot to research the intrusion techniques of malicious actors. This document is broken into the sections necessary to implement the honeypot and process the data logs. Section 2 outlines how to configure the cloud environment. The next section shows the steps to create the virtual machines. The third section provides the steps to install the honeypot Cowrie. Section 4 shows the steps to collect the logs. The final section provides the code used to process the data logs.

## 2 Configuring Cloud Environment

Droplets were set up on Digital Ocean which will each contain a virtual machine to host one honeypot.

List of created Droplets:

The screenshot shows the DigitalOcean dashboard. On the left sidebar, under the 'PROJECTS' section, three projects are listed: 'first-project', 'cowrie with fak...', and 'cowrie top 100 ...'. The 'cowrie with fak...' project is highlighted with a red box. The main content area shows the details for the 'cowrie with fake files' project, which is described as a 'Class project / Educational purposes'. It shows one droplet named 'bob' with a public IP of 209.38.16.42. Below the droplet list, there is a 'Best Practices for Your Data' section with two recommendations: 'Mount a block storage volume' and 'Enable automatic backups'.

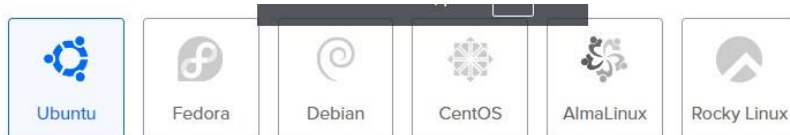
### 3 Creating Virtual Machines

Three virtual machines were set up. Below are the steps to create a virtual machine hosted on Digital Ocean.

Selecting Region:

The screenshot shows the DigitalOcean 'Choose Region' and 'Datacenter' selection screen. The 'Choose Region' section displays a grid of regions with their respective flags: New York, San Francisco, Amsterdam, Singapore, London, Frankfurt, Toronto, Bangalore, and Sydney. The 'Datacenter' section shows a dropdown menu with 'Amsterdam • Datacenter 3 • AMS3' selected. Below the dropdown, there is a tip: 'Tip: Select the datacenter closest to you or your users'. At the bottom, the pricing is shown as '\$6.00/month' and '\$0.009/hour'. There are two buttons: 'CREATE VIA COMMAND LINE' and 'Create Droplet'.

Creating Ubuntu Virtual Machines



Version

24.04 (LTS) x64

Choose Size

Need help picking a plan? [Help me choose](#)

Droplet Type

SHARED CPU	DEDICATED CPU			
Basic (Plan selected)	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized

Selecting CPU type:

CPU options

☒ Regular  
Disk type: SSD

☐ Premium Intel  
Disk: NVMe SSD

☐ Premium AMD  
Disk: NVMe SSD

\$6/mo \$0.009/hour	\$12/mo \$0.018/hour	\$18/mo \$0.027/hour	\$24/mo \$0.036/hour	\$48/mo \$0.071/hour	\$96/mo \$0.143/hour
1 GB / 1 CPU 25 GB SSD Disk 1000 GB transfer	2 GB / 1 CPU 50 GB SSD Disk 2 TB transfer	2 GB / 2 CPUs 60 GB SSD Disk 3 TB transfer	4 GB / 2 CPUs 80 GB SSD Disk 4 TB transfer	8 GB / 4 CPUs 160 GB SSD Disk 5 TB transfer	16 GB / 8 CPUs 320 GB SSD Disk 6 TB transfer

Show all plans

\$6.00/month

\$0.009/hour

[CREATE VIA COMMAND LINE](#)

Create Droplet

Setting root password:

☐ SSH Key

Connect to your Droplet with an SSH key pair

☒ Password

Connect to your Droplet as the "root" user via password

Create root password \*

.....

👁

PASSWORD REQUIREMENTS

✓ Must be at least 8 characters long

✓ Must contain 1 uppercase letter (cannot be first or last character)

✓ Must contain 1 number

✓ Cannot end in a number or special character

\$6.00/month

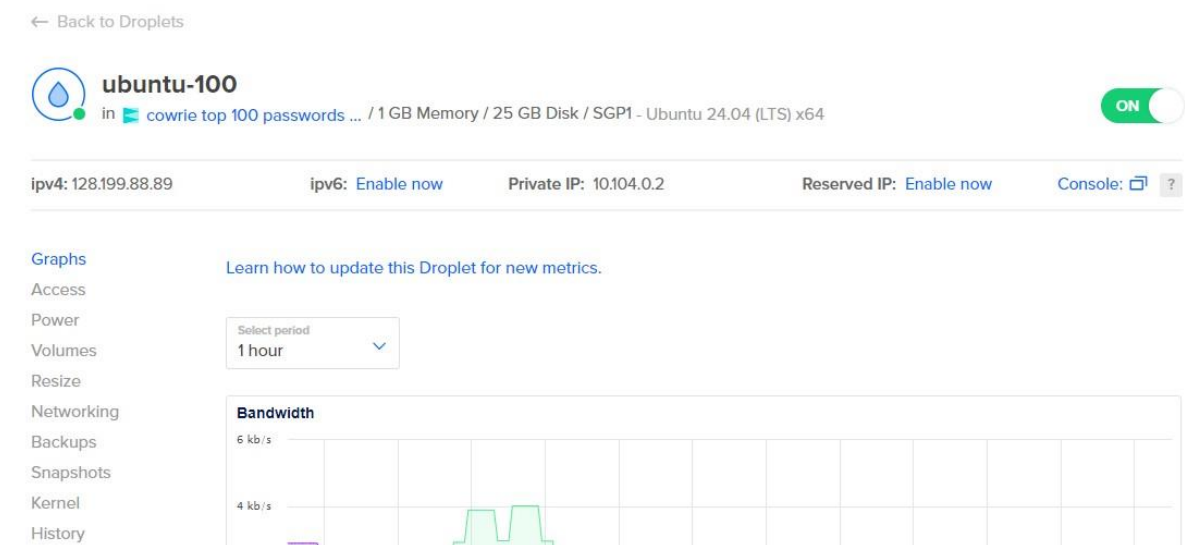
\$0.009/hour

[CREATE VIA COMMAND LINE](#)



Create Droplet

Created Virtual Machines:


128.199.88.89:



209.38.16.42:

 **bob**  
in  cowrie with fake files / 1 GB Memory / 25 GB Disk / SYD1 - Ubuntu 24.04 (LTS) x64

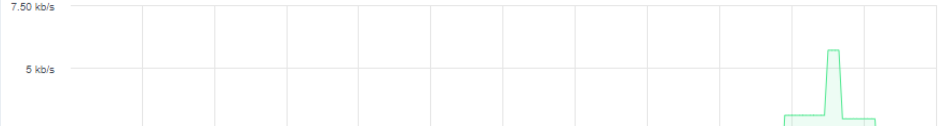
[Upgrade Droplet](#) ON

ipv4: 209.38.16.42    ipv6: [Enable now](#)    Private IP: 10.126.0.2    Reserved IP: [Enable now](#)    Console:  ?

[Graphs](#)  
[Access](#)  
[Power](#)  
[Volumes](#)  
[Resize](#)  
[Networking](#)  
[Backups](#)  
[Snapshots](#)  
[Kernel](#)  
[History](#)



[Learn how to update this Droplet for new metrics.](#)

Select period  
1 hour


**Bandwidth**  


167.71.232.56:

[← Back to Droplets](#)

 **tommy-fs**  
in  cowrie with new system co... / 1 GB Memory / 25 GB Disk / BLR1 - Ubuntu 24.04 (LTS) x64

[Upgrade Droplet](#) ON

ipv4: 167.71.232.56    ipv6: [Enable now](#)    Private IP: 10.122.0.2    Reserved IP: [Enable now](#)    Console:  ?

[Graphs](#)  
[Access](#)  
[Power](#)  
[Volumes](#)  
[Resize](#)  
[Networking](#)  
[Backups](#)  
[Snapshots](#)  
[Kernel](#)  
[History](#)

[Learn how to update this Droplet for new metrics.](#)

Select period  
1 hour

**Bandwidth**  


2024-07-31 18:47:45  
private - inbound: 0 b/s

Connecting To Virtual Machines from personal device.

```
root@bob: ~  
Microsoft Windows [Version 10.0.19045.4651]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Darragh>ssh -p 22 root@209.38.16.42  
root@209.38.16.42's password:  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/pro  
  
System information as of Wed Jul 31 19:04:55 UTC 2024  
  
System load:  0.12           Processes:            101  
Usage of /:   25.7% of 23.17GB Users logged in:          0  
Memory usage: 45%           IPv4 address for eth0: 209.38.16.42  
Swap usage:   0%            IPv4 address for eth0: 10.49.0.5  
  
Expanded Security Maintenance for Applications is not enabled.  
  
40 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
*** System restart required ***  
Pending kernel upgrade!  
Running kernel version:  
  6.8.0-31-generic  
Diagnostics:  
  The currently running kernel version is not the expected kernel version 6.8.0-39-generic.  
Last login: Sun Jun  9 17:28:18 2024 from 213.202.173.219  
root@bob:~# ^C  
root@bob:~# hostname  
bob  
root@bob:~#
```



root@tommy-fs: ~

C:\Users\Darragh>ssh -p 22 root@167.71.232.56

root@167.71.232.56's password:

Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic x86\_64)

\* Documentation: <https://help.ubuntu.com>  
\* Management: <https://landscape.canonical.com>  
\* Support: <https://ubuntu.com/pro>

System information as of Wed Jul 31 19:07:20 UTC 2024

System load:	0.0	Processes:	99
Usage of /:	42.7% of 23.17GB	Users logged in:	0
Memory usage:	45%	IPv4 address for eth0:	167.71.232.56
Swap usage:	0%	IPv4 address for eth0:	10.47.0.5

Expanded Security Maintenance for Applications is not enabled.

36 updates can be applied immediately.

To see these additional updates run: `apt list --upgradable`

Enable ESM Apps to receive additional future security updates.

See <https://ubuntu.com/esm> or run: `sudo pro status`

\*\*\* System restart required \*\*\*

Pending kernel upgrade!

Running kernel version:

6.8.0-31-generic

Diagnostics:

The currently running kernel version is not the expected kernel version 6.8.0-39-generic.

Last login: Tue May 28 20:59:53 2024 from 78.19.241.80

root@tommy-fs:~# hostname

tommy-fs

root@tommy-fs:~#

```
root@ubuntu-100: ~  
C:\Users\Darragh>ssh -p 22 root@128.199.88.89  
root@128.199.88.89's password:  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-35-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/pro  
  
System information as of Wed Jul 31 19:08:54 UTC 2024  
  
System load:  0.08          Processes:            102  
Usage of /:   22.0% of 23.17GB  Users logged in:    0  
Memory usage: 50%          IPv4 address for eth0: 128.199.88.89  
Swap usage:   0%           IPv4 address for eth0: 10.15.0.5  
  
Expanded Security Maintenance for Applications is not enabled.  
  
40 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
*** System restart required ***  
Pending kernel upgrade!  
Running kernel version:  
  6.8.0-35-generic  
Diagnostics:  
  The currently running kernel version is not the expected kernel version 6.8.0-39-generic.  
Last login: Sat Jun 22 10:44:30 2024 from 78.17.129.38  
root@ubuntu-100:~# hostname  
ubuntu-100  
root@ubuntu-100:~#
```

## 4 Installing Cowrie

Start by updating systems

Sudo apt-get upgrade Sudo apt-get update
---

```
root@ubuntu-s-1vcpu-1gb-ams3-01: ~
* Documentation: https://help.ubuntu.com
* Management:   https://landscape.canonical.com
* Support:      https://ubuntu.com/pro

System information as of Mon Jul 1 19:19:50 UTC 2024

System load: 0.0          Processes:              106
Usage of /:  7.1% of 23.17GB Users logged in:                0
Memory usage: 19%        IPv4 address for eth0: 146.190.224.126
Swap usage:  0%          IPv4 address for eth0: 10.18.0.5

Expanded Security Maintenance for Applications is not enabled.

7 updates can be applied immediately.
7 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ubuntu-s-1vcpu-1gb-ams3-01:~# sudo apt-get upgrade
```

Installing git and these Python packages: python3-virtualenv, libssl-dev, libffi-dev, build-essential, libpython3-dev, python3-minimal, authbind and virtualenv

```
sudo apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv
```

```
root@ubuntu-s-1vcpu-1gb-ams3-01:~# sudo apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv
```

Installing python3 virtual environment

```
apt install python3.10-venv
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ubuntu-s-1vcpu-1gb-ams3-01:~# apt install python3.10-venv
```

Installing convenience package to point at python interpreter

```
sudo apt-get install python-is-python3
```

```
root@ubuntu-s-1vcpu-1gb-ams3-01: ~  
root@ubuntu-s-1vcpu-1gb-ams3-01:~# sudo apt-get install python-is-python3
```

Creating Cowrie user with password disabled

```
sudo adduser --disabled-password cowrie
```

```
root@ubuntu-s-1vcpu-1gb-ams3-01:~# sudo adduser --disabled-password cowrie  
info: Adding user `cowrie' ...  
info: Selecting UID/GID from range 1000 to 59999 ...  
info: Adding new group `cowrie' (1000) ...  
info: Adding new user `cowrie' (1000) with group `cowrie (1000)' ...  
info: Creating home directory `/home/cowrie' ...  
info: Copying files from `/etc/skel' ...  
Changing the user information for cowrie  
Enter the new value, or press ENTER for the default  
    Full Name []:  
    Room Number []:  
    Work Phone []:  
    Home Phone []:  
    Other []:  
Is the information correct? [Y/n] y  
info: Adding new user `cowrie' to supplemental / extra groups `users' ...  
info: Adding user `cowrie' to group `users' ...  
root@ubuntu-s-1vcpu-1gb-ams3-01:~#
```

Switching to Cowrie user

```
su - cowrie
```

```
root@ubuntu-s-1vcpu-1gb-ams3-01:~#  
root@ubuntu-s-1vcpu-1gb-ams3-01:~# su - cowrie  
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$
```

Cloning Cowrie git repository

```
git clone http://github.com/cowrie/cowrie
```

```

cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$ git clone http://github.com/cowrie/cowrie
Cloning into 'cowrie'...
warning: redirecting to https://github.com/cowrie/cowrie/
remote: Enumerating objects: 17602, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 17602 (delta 30), reused 41 (delta 15), pack-reused 17531
Receiving objects: 100% (17602/17602), 9.94 MiB | 17.27 MiB/s, done.
Resolving deltas: 100% (12354/12354), done.
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$

```

Switching to cowrie directory and installing Python pip

```

cd cowrie
python -m pip install --upgrade pippython -m pip install --upgrade pippython -m pip install
--upgrade pip

```

```

cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~$ cd cowrie
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ python -m pip install --upgrade pippython -m pip install --upgrade pippython -m pip install --upgrade pip

```

Installing Python3-pip module

```

Sudo apt install python3-pip

```

```

root@ubuntu-s-1vcpu-1gb-ams3-01:~$
root@ubuntu-s-1vcpu-1gb-ams3-01:~# sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done

```

Installing pip

```

python -m pip install --upgrade pip

```

```

cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ python -m pip install --upgrade pip

```

Installing pip package and recommendations from requirements text.

```
python -m pip install --upgrade -r requirements.txt
```

```
hint: See PEP 668 for the detailed specification.  
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ python -m pip install --upgrade -r requirements.txt
```

## Copying Config files

```
cd etc  
cp cowrie.cfg.dist cowrie.cfg
```

```
hint: See PEP 668 for the detailed specification.  
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ cd etc  
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie/etc$ cp cowrie.cfg.dist cowrie.cfg  
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie/etc$
```

## Changing hostname of cowrie server

```
nano cowrie.cfg
```

```
GNU nano 7.2  
  
# Hostname for the honeypot. Displayed by the shell prompt  
# environment  
#  
# (default: svr04)  
hostname = joes server  
  
# Directory where to save log files in.  
#  
# (default: log)  
log_path = var/log/cowrie  
  
# Directory where to save downloaded artifacts in.  
#  
# (default: downloads)  
download_path = ${honeypot:state_path}/downloads  
  
# Directory for static data files  
#  
# (default: share/cowrie)  
share_path = share/cowrie
```



## Enabling Telnet

```
# Configure keyboard-interactive login
auth_keyboard_interactive_enabled = false

# =====
# Telnet Specific Options
# =====
[telnet]

# Enable Telnet support, disabled by default
enabled = true

# Endpoint to listen on for incoming Telnet connections.
# See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
# (default: listen_endpoints = tcp:2223:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
```

## Changing passwd file

```
cd honeyfs/etc
nano passwd
```

```
cd...: command not found
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie/et$ cd ..
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie$ cd honeyfs/etc
-bash: cd: honeyfs/etc: No such file or directory
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie$ ls
CHANGELOG.rst  INSTALL.rst  MANIFEST.in  README.rst  docker  etc  pyproject.toml  requirements-output.txt  setup.cfg  share  tox.ini
CONTRIBUTING.rst  LICENSE.rst  Makefile  bin  docs  honeyfs  requirements-dev.txt  requirements.txt  setup.py  src  var
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie$ cd honeyfs/etc
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie/honeyfs/etc$ nano passwd
```

## Blocking top 100 passwords on Honeypot A – 128.199.88.89

```
*:x:!12345
*:x:!1234567890
*:x:!senha
*:x:!1234567
*:x:!qwerty
*:x:!abc123
*:x:!Million2
*:x:!0
*:x:!1234
*:x:!iloveyou
*:x:!aaron431
*:x:!password1
*:x:!qqww1122
*:x:!123
*:x:!omgpop
*:x:!123321
*:x:!654321
*:x:!qwertyuiop
*:x:!qwert123456
*:x:!123456a
*:x:!a123456
*:x:!666666
*:x:!asdfghjkl
*:x:!ashley
*:x:!987654321
*:x:!unknown
*:x:!zxcvbnm
*:x:!112233
*:x:!chatbooks
*:x:!20100728
*:x:!123123123
*:x:!princess
*:x:!jacket025
*:x:!evite
*:x:!123abc
*:x:!123qwe
*:x:!sunshine
*:x:!121212
*:x:!dragon
*:x:!1q2w3e4r
```

## Changing default Cowrie user name

```
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie/honeyfs/etc
GNU nano 7.2
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
joe:x:1000:1000:Phil California,,,:/home/phil:/bin/bash
```

## Creating fake directory

```
bin/fsctl share/cowrie/fs.pickle
share/cowrie/fs.pickle
fs.pickle/$ mv /home/phil /home/rohit
exit
```

```
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie/honeyfs/etc$ cd ..
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie/honeyfs$ cd ..
cowrie@ubuntu-s-1vcpu-1gb-ams3-01: ~/cowrie$ bin/fsctl share/cowrie/fs.pickle
share/cowrie/fs.pickle

Kippo/Cowrie file system interactive editor
Donovan Hubbard, Douglas Hubbard, March 2013
Type 'help' for help

fs.pickle/$ mv /home/phil /home/rohit
File moved from /home/phil to /home/rohit
fs.pickle/$
```

## Creating fake files



```

cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie/home$ cd ..
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ bin/fsctl share/
share/cowrie/fs.pickle

Kippo/Cowrie file system interactive editor
Donovan Hubbard, Douglas Hubbard, March 2013
Type 'help' for help

fs.pickle:/$ mv /home/phil /home/rohit
File moved from /home/phil to /home/rohit
fs.pickle:/$ touch passwords.txt
Added '/passwords.txt'
fs.pickle:/$

```

```

Kippo/Cowrie file system interactive editor
Donovan Hubbard, Douglas Hubbard, March 2013
Type 'help' for help

fs.pickle:/$ mv /home/phil /home/rohit
File moved from /home/phil to /home/rohit
fs.pickle:/$ touch passwords.txt
Added '/passwords.txt'
fs.pickle:/$ touch invoices
Added '/invoices'
fs.pickle:/$ touch bill.txt
Added '/bill.txt'
fs.pickle:/$

```

Creating Cowrie virtual environment

```
python -m venv cowrie-env
```

```

cowrie@ubuntu-s-1vcpu-1gb-ams3-01:~/cowrie$ cd ../../
cowrie@ubuntu-s-1vcpu-1gb-ams3-01:/home$ python -m venv cowrie-env

```

Starting Cowrie virtual environment

```
source cowrie-env/bin/activate
```

```

cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$ source cowrie-env/bin/activate
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$

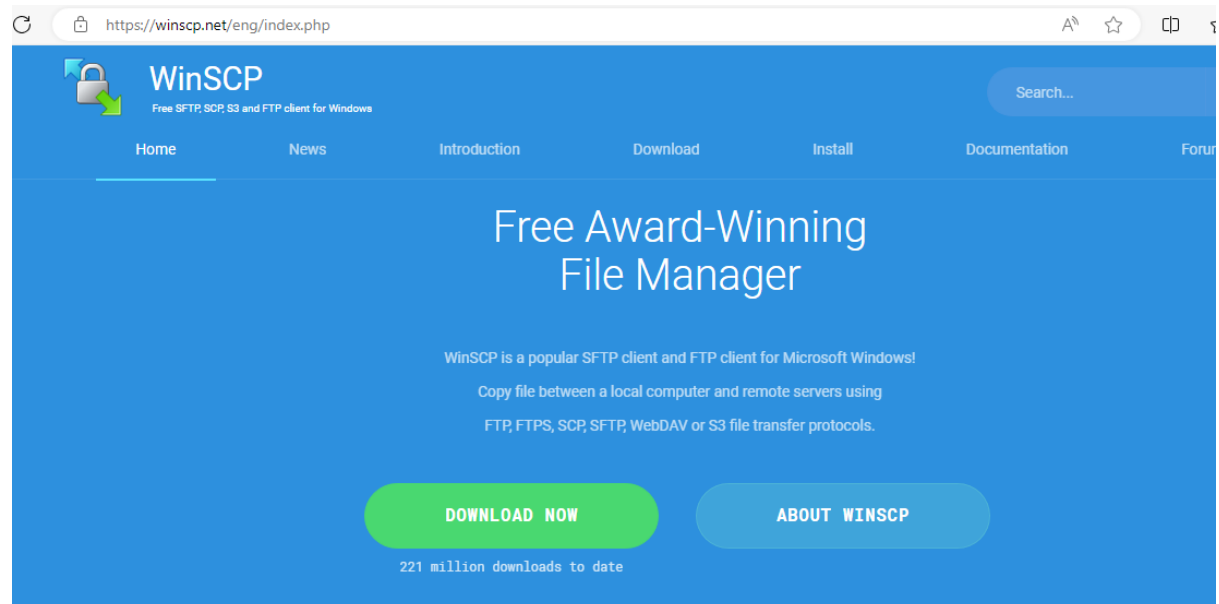
```

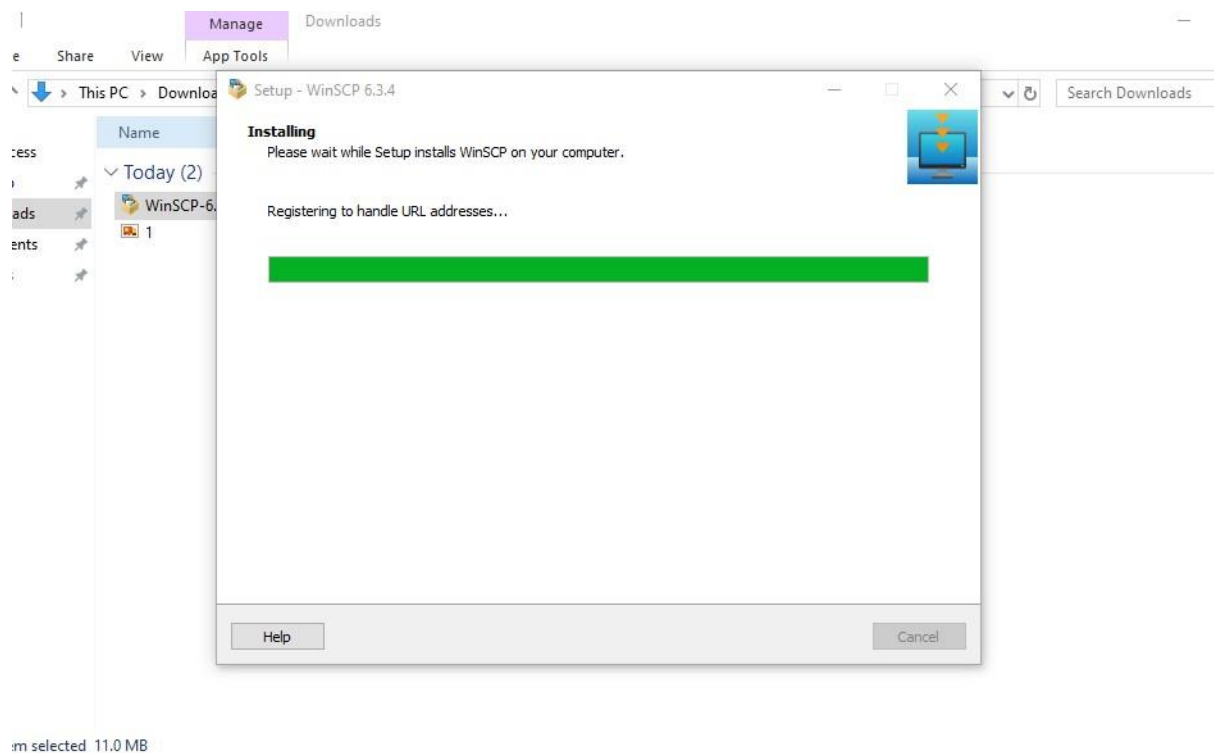
```
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$ bin/cowrie status  
cowrie is running (PID: 1615).  
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$
```

```
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$  
(cowrie-env) cowrie@ubuntu-s-1vcpu-2gb-lon1-01:~/cowrie$ bin/cowrie start  
Using activated Python virtual environment "/home/cowrie/cowrie/cowrie-env"
```

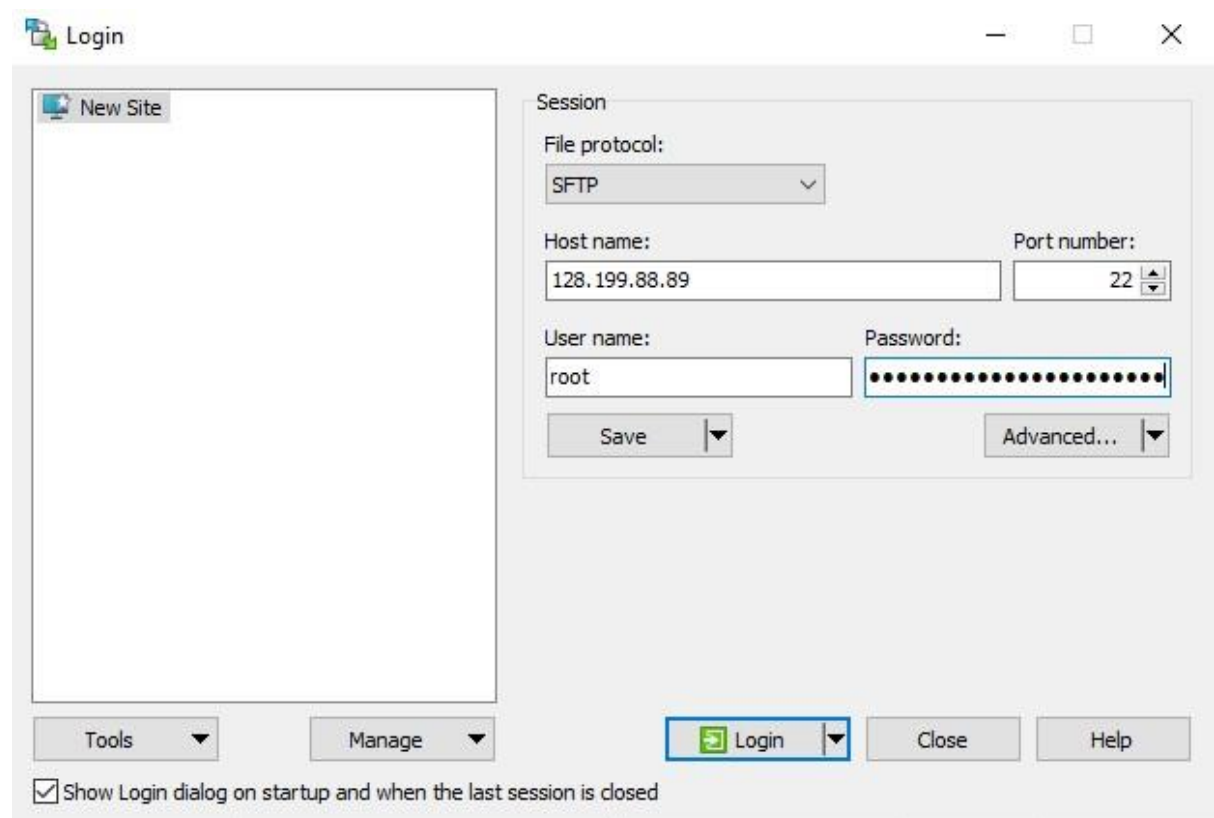
## 5 Collecting Logs

### Installing WinSCP

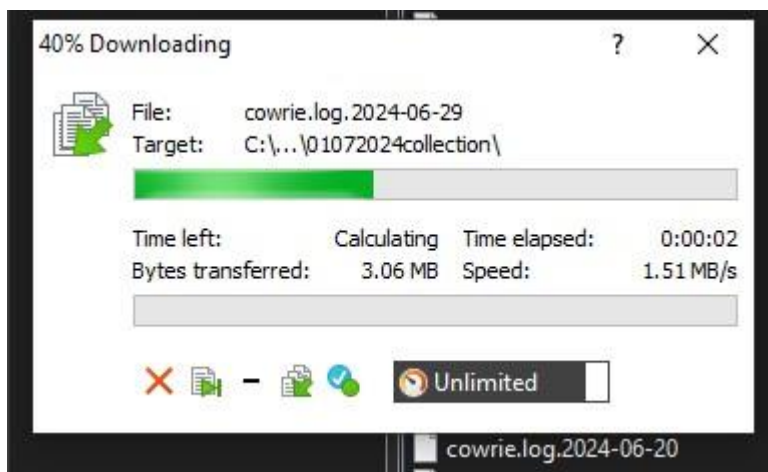
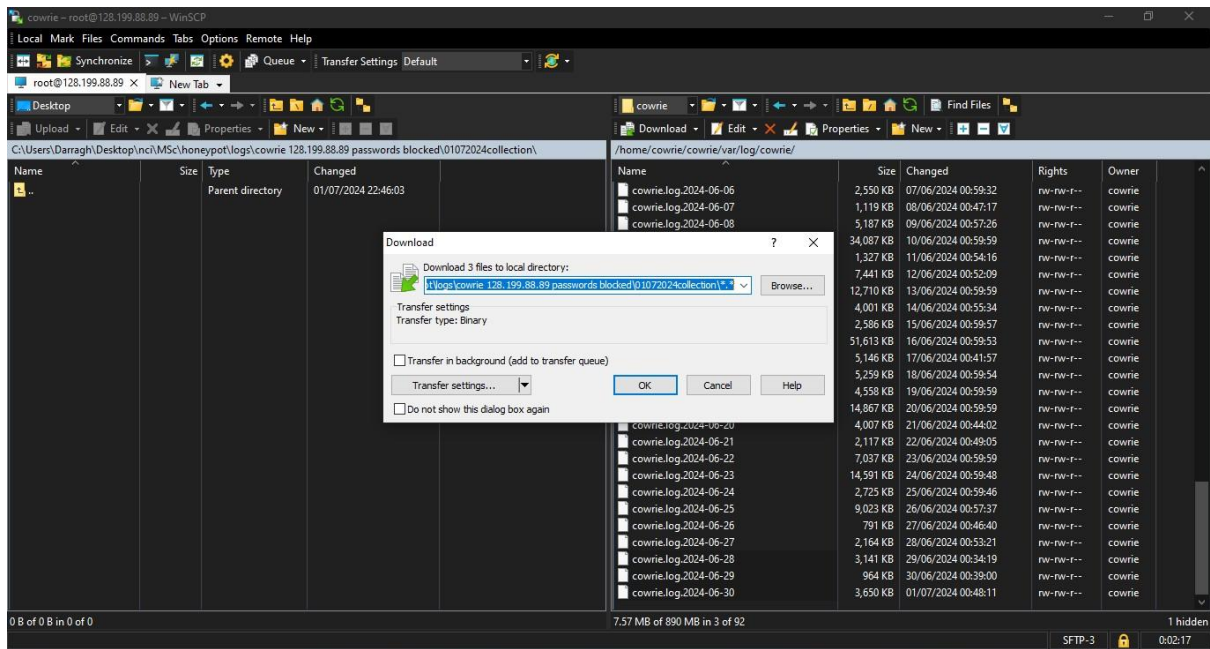




## Connecting to honeypot



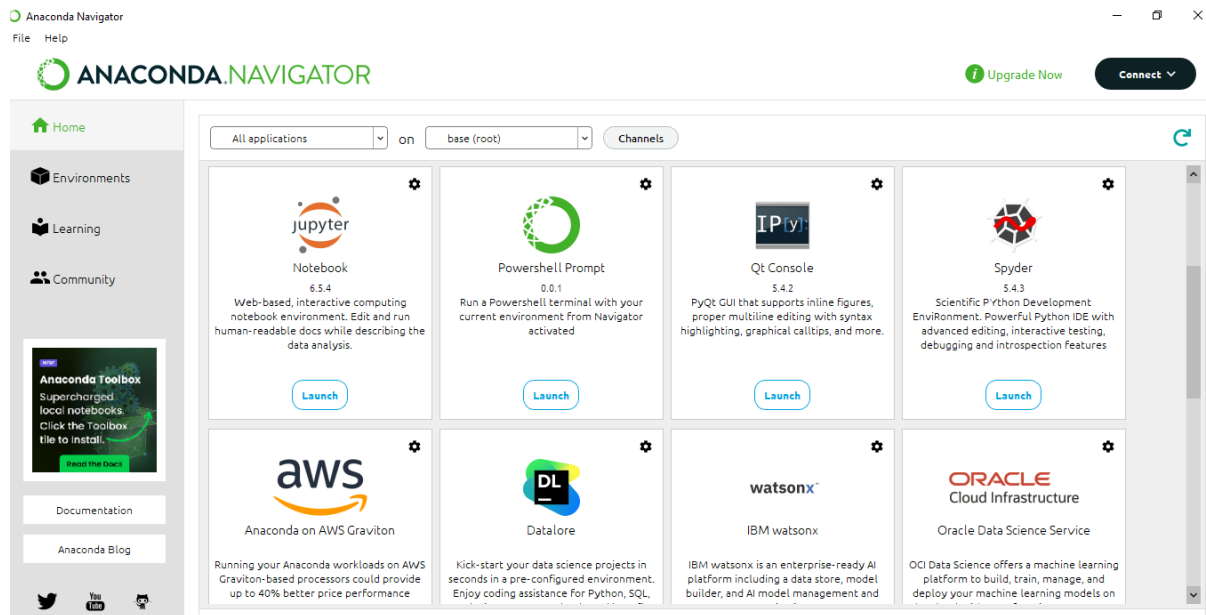
## Downloading files



## 6 Processing Logs

### Python Code

Anaconda and Jupyter notebook was used to run the Python scripts



## Script to split logs into smaller files

```

1 def split_file(input_file, output_folder, lines_per_file):
2     import os
3
4     if not os.path.exists(output_folder):
5         os.makedirs(output_folder)
6
7     with open(input_file, 'r', encoding="utf8") as f:
8         part_number = 0
9         lines = []
10
11     for line in f:
12         lines.append(line)
13         if len(lines) >= lines_per_file:
14             part_number += 1
15             output_file = os.path.join(output_folder, f'part_{part_number}.txt')
16             with open(output_file, 'w', encoding="utf8") as out_f:
17                 out_f.writelines(lines)
18                 lines = []
19
20     # Write remaining lines
21     if lines:
22         part_number += 1
23         output_file = os.path.join(output_folder, f'part_{part_number}.txt')
24         with open(output_file, 'w', encoding="utf8") as out_f:
25             out_f.writelines(lines)
26             print("complete")
27
28 input_file = './cowrie.log.2024-06-01.txt'
29 output_folder = 'C:\\Users\\Darragh\\cowrie log analysis\\oneMonthLogs\\a - 128.199.88.89 cowrie with 100 top passwords bloc
30 lines_per_file = 10000 # Adjust the number of lines per split file
31
32 split_file(input_file, output_folder, lines_per_file)
33

```

## Script1 counts ips and exports into dictionary in desc order

This script iterates through each of the files and finds patterns that match new IP connections. It then counts each instance of an IP and exports them into a text file giving a total for each address.

```

1 import os
2 import re
3 onlyips = []
4 #get data
5 paragraphs = []
6 #num = 1
7
8 num+=1
9
10
11 #collects new connections
12 num1 = 0
13 count = 0
14 newConn = []
15 myArray = []
16
17
18 #create array of only IPs from new connections
19 #####
20 while os.path.exists('./part_'+ str(num) + '.txt'):
21     with open('./part_'+ str(num) + '.txt', encoding='utf-8') as f:
22         paragraphs = f.readlines()
23         print("END...")
24         myArray.append(paragraphs)
25         num+=1
26         #print(paragraphs)
27 else:
28     print("not found")
29
30 myString = ', '.join(map(str, myArray))
31
32
33 #pattern = r'(\d+\.\d+\.\d+\.\d+)'
34 pattern = r"New connection: ([\d.]+):\d+"
35
36 matches = re.findall(pattern, myString)
37
38 with open('bunch of ips.txt', 'w') as file:
39     # Join the array elements with a comma and write to the file
40     file.write(str(matches))
41
42 print("Complete")
43
44 #unique IPs
45 myset = set(matches)
46 print(myset)
47
48 my_dict = {item: 0 for item in myset}
49
50 for x in matches:
51     print(x)
52     if x in my_dict:
53         my_dict.update({x: my_dict[x]+1})
54
55
56
57 sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))
58
59
60 with open('counted ips in dictionary.txt', 'w') as file:
61     # Join the array elements with a comma and write to the file
62     file.write(str(sorted_dict))
63     print("complete")

```

```

import os
import re
onlyips = []
#get data
paragraphs = []
#num = 1

num+=1

```

```

#collects new connections
num1 = 0
count = 0
newConn = []
myArray = []

#create array of only IPs from new connections
#####
while os.path.exists('./part_' + str(num) + '.txt'):
    with open('./part_' + str(num) + '.txt', encoding='utf-8') as f:
        paragraphs = f.readlines()
        print("END...")
        myArray.append(paragraphs)
        num+=1
        #print(paragraphs)
else:
    print("not found")

myString = ','.join(map(str, myArray))

#pattern = r'(\d+\.\d+\.\d+\.\d+)'
pattern = r"New connection: ([\d.]+):\d+"

matches = re.findall(pattern, myString)

with open('bunch of ips.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(str(matches))

print("Complete")

#unique IPs
myset = set(matches)
print(myset)

my_dict = {item: 0 for item in myset}

for x in matches:
    print(x)
    if x in my_dict:
        my_dict.update({x: my_dict[x]+1})

sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))

```

```
with open('counted ips in dictionary.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(str(sorted_dict))
    print("complete")
```

Script2 extract passwords and export to text file

The script iterates through each file and finds a pattern that matches the inputted password. The passwords are then exported to a text file.

```
: 1 import os
2 import re
3 num = 1
4 paragraphs = []
5 num1 = 0
6 count = 0
7 newConn = ["1"]
8 passwords = []
9 #pattern = r"/b'([^\']*)'"
10 #pattern = r"'/b'([^\']+)'"
11 pattern = r"/b'(.*)'"
12
13 while os.path.exists('./part_' + str(num) + '.txt'):
14     with open('./part_' + str(num) + '.txt', encoding='utf-8') as f:
15         paragraphs = f.readlines()
16         num+=1
17
18
19     for x in range(0, len(paragraphs)):
20         matches = re.findall(pattern, paragraphs[x])
21         if matches:
22
23             # Convert matches list to a comma-separated string without brackets
24             matches_str = ', '.join(matches)
25             print(matches_str)
26             with open("password collection.txt", "a") as file:
27                 #file.write(matches_str + "\n")
28                 file.write(matches_str + ",")
```

```
import os
import re
num = 1
paragraphs = []
num1 = 0
count = 0
newConn = ["1"]
passwords = []
#pattern = r"/b'([^\']*)'"
#pattern = r"'/b'([^\']+)'"
pattern = r"/b'(.*)'"
```



```
while os.path.exists('./part_'+ str(num) + '.txt'):
    with open('./part_'+ str(num) + '.txt', encoding='utf-8') as f:
        paragraphs = f.readlines()
        num+=1

    for x in range(0, len(paragraphs)):
        matches = re.findall(pattern, paragraphs[x])
        if matches:

            # Convert matches list to a comma-separated string without brackets
            matches_str = ', '.join(matches)
            print(matches_str)
            with open("password collection.txt", "a") as file:
                #file.write(matches_str + "\n")
                file.write(matches_str + ",")
```

script3 count passwords and export dictionary

The script opens the passwords file and counts the number of occurrences of each password and exports the total count numbers into a text file.

```

1 import os
2 import re
3 num = 1
4 paragraphs = []
5 num1 = 0
6 count = 0
7 newConn = ["1"]
8 passwords = []
9 #pattern = r"/b'([^\']*)'"
10 #pattern = r"/b'([^\']+)'"
11 pattern = r"/b'(.*)'"
12
13 #while os.path.exists('password collection.txt'):
14 with open('password collection.txt', encoding='utf-8') as f:
15     paragraphs = f.readlines()
16     num+=1
17
18
19 for x in range(0, len(paragraphs)):
20     #matches = re.findall(pattern, paragraphs[x])
21     #if matches:
22         # Convert matches list to a comma-separated string without brackets
23         #matches_str = ', '.join(matches)
24         print(paragraphs)
25         #with open("password collection.txt", "a") as file:
26             #file.write(matches_str + "\n")
27
28 x = str(paragraphs[0])
29 array = x.split(',')
30 myset = set(array)
31
32 my_dict = {item: 0 for item in myset}
33
34 for x in array:
35     print(x)
36     if x in my_dict:
37         my_dict.update({x: my_dict[x]+1})
38
39 sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))
40
41 with open('counted passwords in dictionary.txt', 'w') as file:
42     # Join the array elements with a comma and write to the file
43     file.write(str(sorted_dict))

```

```

import os
import re
num = 1
paragraphs = []
num1 = 0
count = 0
newConn = ["1"]
passwords = []
#pattern = r"/b'([^\']*)'"
#pattern = r"/b'([^\']+)'"
pattern = r"/b'(.*)'"

#while os.path.exists('password collection.txt'):

```

```

with open('password collection.txt', encoding='utf-8') as f:
    paragraphs = f.readlines()
    num+=1

for x in range(0, len(paragraphs)):
    #matches = re.findall(pattern, paragraphs[x])
    #if matches:
    #    # Convert matches list to a comma-separated string without brackets
    #    matches_str = ','.join(matches)
    #    print(paragraphs)
    #    #with open("password collection.txt", "a") as file:
    #        #file.write(matches_str + "\n")

x = str(paragraphs[0])
array = x.split(',')
myset = set(array)

my_dict = {item: 0 for item in myset}

for x in array:
    print(x)
    if x in my_dict:
        my_dict.update({x: my_dict[x]+1})

sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))

with open('counted passwords in dictionary.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(str(sorted_dict))

```

Script4 timestamps exported with commands

Script finds patterns that matches a timestamp and exports them into a text file. Initially planned to be used to identify the hours with the most activity.

```

1 import re
2 import os
3 #get data
4 paragraphs = []
5 num+=1
6 onlyips = []
7
8 #collects new connections
9 num1 = 0
10 count = 0
11 newConn = []
12 myArray = []
13
14 while os.path.exists('./part_'+ str(num) + '.txt'):
15     with open('./part_'+ str(num) + '.txt', encoding='utf-8') as f:
16         paragraphs = f.readlines()
17         print("END...")
18         myArray.append(paragraphs)
19         num+=1
20 else:
21     print("not found")
22
23 myString = ','.join(map(str, myArray))
24
25 pattern = r'(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{6}Z) \[.*?\] NEW KEYS'
26
27 matches = re.findall(pattern, myString)
28
29 #unique IPs
30 myset = set(matches)
31
32 my_dict = {item: 0 for item in myset}
33
34 for x in matches:
35     print(x)
36     if x in my_dict:
37         my_dict.update({x: my_dict[x]+1})
38
39 sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))
40
41 with open('timestamps for new connections.txt', 'w') as file:
42     # Join the array elements with a comma and write to the file
43     file.write(str(matches))

```

```

import re
import os
#get data
paragraphs = []
num+=1
onlyips = []

#collects new connections
num1 = 0
count = 0
newConn = []
myArray = []

while os.path.exists('./part '+ str(num) + '.txt'):

```

```

with open('./part_' + str(num) + '.txt', encoding='utf-8') as f:
    paragraphs = f.readlines()
    print("END...")
    myArray.append(paragraphs)
    num+=1
else:
    print("not found")

myString = ','.join(map(str, myArray))

pattern = r'(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{6}Z) \[.*?\] NEW KEYS'

matches = re.findall(pattern, myString)

#unique IPs
myset = set(matches)

my_dict = {item: 0 for item in myset}

for x in matches:
    print(x)
    if x in my_dict:
        my_dict.update({x: my_dict[x]+1})

sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1], reverse=True))

with open('timestamps for new connections.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(str(matches))
    print("complete")

```

Script5 count and print total commands

This script is used to find matching patterns and then count and export the total to a text file. The pattern is changed to match various commands or files such as 'uname' or 'redtail'.

```

1 import re
2 import os
3 #get data
4 paragraphs = []
5 onlyips = []
6 num+=1
7
8 #collects new connections
9 num1 = 0
10 count = 0
11 newConn = []
12 myArray = []
13
14 #create array of only IPs from new connections
15 #####
16 while os.path.exists('./part_'+ str(num) + '.txt'):
17     with open('./part_'+ str(num) + '.txt', encoding='utf-8') as f:
18         paragraphs = f.readlines()
19         print("END...")
20         myArray.append(paragraphs)
21     num+=1
22     #print(paragraphs)
23 else:
24     print("not found")
25
26 myString = ','.join(map(str, myArray))
27
28 pattern = r'(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{6}Z) \[.*?\] NEW KEYS'
29
30 matches = re.findall(pattern, myString)
31 if matches:
32     print("yeah")
33
34 with open('count for new e ok command.txt', 'w') as file:
35     # Join the array elements with a comma and write to the file
36     file.write(f"The command was found {str(len(matches))} times")
37     print("complete")

```

```

import re
import os
#get data
paragraphs = []
onlyips = []
num+=1

#collects new connections
num1 = 0
count = 0
newConn = []
myArray = []

#create array of only IPs from new connections
#####

```

```

while os.path.exists('./part_'+ str(num) + '.txt'):
    with open('./part_'+ str(num) + '.txt', encoding='utf-8') as f:
        paragraphs = f.readlines()
        print("END...")
        myArray.append(paragraphs)
    num+=1
    #print(paragraphs)
else:
    print("not found")

myString = ','.join(map(str, myArray))

pattern = r'(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{6}Z) \[.*?\] NEW KEYS'

matches = re.findall(pattern, myString)
if matches:
    print("yeah")

with open('count for new e ok command.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(f'The command was found {str(len(matches))} times')
    print("complete")

```

Script6 various commands and print total

This script is used to find matching patterns and then count and export the total to a text file. Multiple patterns were identified:

- Grep name
- Busybox
- Pkill
- uname

```

]: 1 import os
2 import re
3 onlyips = []
4 pattern = r'uname'
5 pattern2 = r'grep name'
6 pattern3 = r'pkill'
7 pattern4 = r'busybox'
8 #get data
9 paragraphs = []
10 num+=1
11 #collects new connections
12 num1 = 0
13 count = 0
14 newConn = []
15 myArray = []
16
17 while os.path.exists('./part_' + str(num) + '.txt'):
18     with open('./part_' + str(num) + '.txt', encoding='utf-8') as f:
19         paragraphs = f.readlines()
20         print("END...")
21         myArray.append(paragraphs)
22         num+=1
23         #print(paragraphs)
24     else:
25         print("not found")
26
27 myString = ','.join(map(str, myArray))
28
29 matches = re.findall(pattern, myString)
30 if matches:
31     print("yeah")
32
33 matches2 = re.findall(pattern2, myString)
34 if matches2:
35     print("yeah")
36
37 matches3 = re.findall(pattern3, myString)
38 if matches3:
39     print("yeah")
40
41 matches4 = re.findall(pattern4, myString)
42 if matches4:
43     print("yeah")
44
45 with open('count for various commands.txt', 'w') as file:
46     # Join the array elements with a comma and write to the file
47     file.write(f"The command uname was found {str(len(matches))} times.\n" + f"The command grep name was found {str(len(mat
48     print("complete")

```

```

import os
import re
onlyips = []
pattern = r'uname'
pattern2 = r'grep name'
pattern3 = r'pkill'
pattern4 = r'busybox'
#get data
paragraphs = []
num+=1
#collects new connections
num1 = 0
count = 0
newConn = []
myArray = []

while os.path.exists('./part_' + str(num) + '.txt'):
    with open('./part_' + str(num) + '.txt', encoding='utf-8') as f:
        paragraphs = f.readlines()
        print("END...")

```



```

        myArray.append(paragraphs)
        num+=1
        #print(paragraphs)
    else:
        print("not found")

myString = ','.join(map(str, myArray))

matches = re.findall(pattern, myString)
if matches:
    print("yeah")

matches2 = re.findall(pattern2, myString)
if matches2:
    print("yeah")

matches3 = re.findall(pattern3, myString)
if matches3:
    print("yeah")

matches4 = re.findall(pattern4, myString)
if matches4:
    print("yeah")

with open('count for various commands.txt', 'w') as file:
    # Join the array elements with a comma and write to the file
    file.write(f"The command uname was found {str(len(matches))} times.\n" + f"The command grep name was found {str(len(matches2))} times.\n" + f"The command pkill was found {str(len(matches3))} times.\n" + f"The command busybox was found {str(len(matches4))} times.\n" )
    #file.write(f"The command grep name found {str(len(matches2))} times")
    print("complete")

```

script 7 import csv capture country export csv

The Ips from 'counted ips in dictionary.txt' file are placed into a file called 'ips.csv' which is iterated through and the origin of country for each IP is returned from the ipinfo api. The results are exported into a csv document.

```

1 import csv
2 import requests
3
4 def get_country_from_ip(ip_address):
5     try:
6         response = requests.get(f'https://ipinfo.io/{ip_address}/json')
7         response.raise_for_status()
8         data = response.json()
9         country = data.get('country')
10        return country
11    except requests.RequestException as e:
12        print(f"Request error for IP {ip_address}: {e}")
13        return None
14    except ValueError:
15        print(f"Error parsing response for IP {ip_address}")
16        return None
17
18 # File paths
19 input_csv_file_path = 'ips.csv'
20 output_csv_file_path = 'ip_countries 209.38.16.42.csv'
21
22 try:
23     # Read the input CSV file
24     with open(input_csv_file_path, mode='r', encoding='utf-8-sig') as infile:
25         csv_reader = csv.DictReader(infile)
26
27         # Debugging: Print the fieldnames to check the headers
28         print(f"CSV headers: {csv_reader.fieldnames}")
29
30         # Ensure the 'ip_address' column is present
31         if 'ip_address' not in csv_reader.fieldnames:
32             raise KeyError("CSV does not contain 'ip_address' column")
33
34         # Prepare to write to the output CSV file
35         with open(output_csv_file_path, mode='w', newline='', encoding='utf-8') as outfile:
36             fieldnames = ['ip_address', 'country']
37             csv_writer = csv.DictWriter(outfile, fieldnames=fieldnames)
38             csv_writer.writeheader()
39
40             # Process each row in the input CSV
41             for row in csv_reader:
42                 ip_address = row['ip_address'].strip()
43                 if ip_address:
44                     country = get_country_from_ip(ip_address)
45                     print(f'IP Address: {ip_address}, Country: {country}')
46                     # Write the result to the output CSV
47                     csv_writer.writerow({'ip_address': ip_address, 'country': country})
48
49 except FileNotFoundError:
50     print(f"File not found: {input_csv_file_path}")
51 except KeyError as e:
52     print(e)
53 except Exception as e:
54     print(f"An unexpected error occurred: {e}")
55

```

```

import csv
import requests

def get_country_from_ip(ip_address):
    try:
        response = requests.get(f'https://ipinfo.io/{ip_address}/json')
        response.raise_for_status()
        data = response.json()
        country = data.get('country')

```

```

        return country
    except requests.RequestException as e:
        print(f'Request error for IP {ip_address}: {e}')
        return None
    except ValueError:
        print(f'Error parsing response for IP {ip_address}')
        return None

# File paths
input_csv_file_path = 'ips.csv'
output_csv_file_path = 'ip_countries 209.38.16.42.csv'

try:
    # Read the input CSV file
    with open(input_csv_file_path, mode='r', encoding='utf-8-sig') as infile:
        csv_reader = csv.DictReader(infile)

        # Debugging: Print the fieldnames to check the headers
        print(f'CSV headers: {csv_reader.fieldnames}')

        # Ensure the 'ip_address' column is present
        if 'ip_address' not in csv_reader.fieldnames:
            raise KeyError("CSV does not contain 'ip_address' column")

        # Prepare to write to the output CSV file
        with open(output_csv_file_path, mode='w', newline='', encoding='utf-8') as outfile:
            fieldnames = ['ip_address', 'country']
            csv_writer = csv.DictWriter(outfile, fieldnames=fieldnames)
            csv_writer.writeheader()

            # Process each row in the input CSV
            for row in csv_reader:
                ip_address = row['ip_address'].strip()
                if ip_address:
                    country = get_country_from_ip(ip_address)
                    print(f'IP Address: {ip_address}, Country: {country}')
                    # Write the result to the output CSV
                    csv_writer.writerow({'ip_address': ip_address, 'country': country})

except FileNotFoundError:
    print(f'File not found: {input_csv_file_path}')
except KeyError as e:
    print(e)
except Exception as e:
    print(f'An unexpected error occurred: {e}')

```