Configuration Manual

MSc Research Project
Cyber Security

Jithin Paul John
Student ID: x20254857

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Jithin Paul Joh ……..……………………………………………………………………………………

Student ID: x20254857 ……………………………………………………………………………………………..……

Programme: MSc Cyber Security ……………………………………………………… **Year:** 2022-2023 …………………………..

Module: MSc Research Project ……………………………………………………………………………………….………

Lecturer: Imran Khan ……………………………………………………………………………………….………

Submission Due Date: 15-12-2022 ……………………………………………………………………………………….………

Project Title: Novel technique for detecting unknown threats using honeypot instead of purple teaming in organization ……………………………………………………………………………………….………

Word Count: 1742 ……………………………………… **Page Count:** 09…………………………………….…….………

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: Jithin Paul John ……………………………………………………………………………………………………

Date: 15-12-2022 ……………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY 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.	☑

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jithin Paul John
Student ID: x20254857

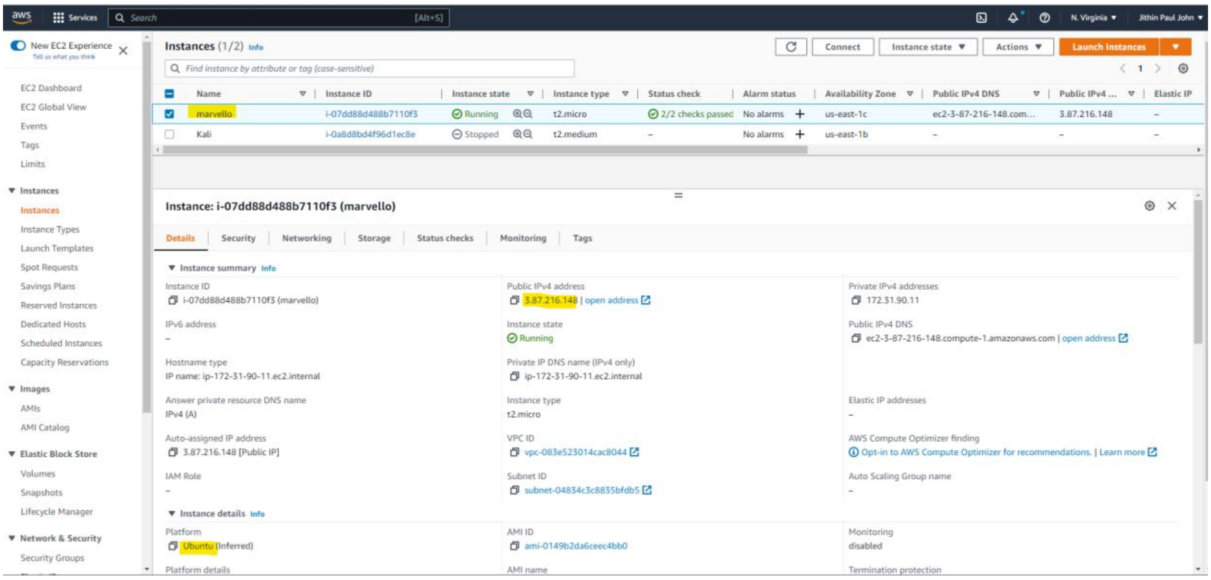
1 Introduction

This project manual includes information about the software, hardware, and other tools used to deploy this research setup. This research employing honeypot, custom IDS for detecting unknown threats relies on the configuration manual for deployment. This contains the overall configuration and commands for performing this operation.

2 Deployment Requirements

The project has been deployed on an Amazon EC2 instance as the deployment is easy compared to a host machine. It also comes with a variety of operating systems as per the requirement. As it has a public IP, it is easier to capture attacks.

Operating System: Ubuntu 22.04
Storage: 8GB



Instances (1/2) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
☑	marvello	i-07dd88d488b7110f3	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-3-87-216-148.com...	3.87.216.148	–
□	Kali	i-0a8d8bd4f96d1ec8e	Stopped	t2.medium	–	No alarms	us-east-1b	–	–	–

Instance: i-07dd88d488b7110f3 (marvello)

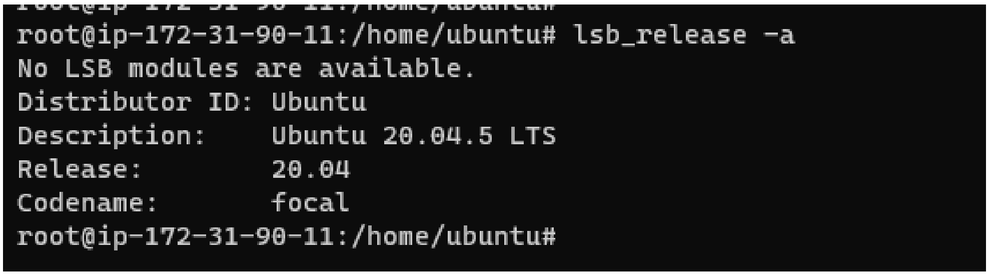
Instance summary Info

Instance ID: i-07dd88d488b7110f3 (marvello)
Public IPv4 address: 3.87.216.148 | open address
Private IPv4 addresses: 172.31.90.11
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-3-87-216-148.compute-1.amazonaws.com | open address
Hostname type: IP name: ip-172-31-90-11.ec2.internal
Private IP DNS name (IPv4 only): ip-172-31-90-11.ec2.internal
Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro
Elastic IP addresses: –
Auto-assigned IP address: 3.87.216.148 [Public IP]
VPC ID: vpc-083e523014cac8044
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: –
Subnet ID: subnet-04834c3c8835bfdb5
Auto Scaling Group name: –

Instance details Info

Platform: Ubuntu (Inferred)
AMI ID: ami-0149b2da6ceec4bb0
Monitoring: disabled
Platform details
AMI name
Termination protection

Figure 1 EC2 Instance details



```
root@ip-172-31-90-11:/home/ubuntu# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.5 LTS
Release:        20.04
Codename:       focal
root@ip-172-31-90-11:/home/ubuntu#
```

Figure 2 OS details

3 Tools Used

Below listed are the tools used in the research for capturing unknown threats.

Python: Version 2.6, 2.7, or 3.x is required
Honeypot: Dionaea
IDS: Custom IDS (Python Programming)
Packet analyser: Pcapy-ng
Database: SQLite
Dashboard: Web application

3.1 Python

Python is a common programming language that is frequently used to create operating system scripts. It is suitable to be used in both web development and app creation. Python is one of the dependencies for custom IDS that is written in Python. Hence, python version 2.6, 2.7, or 3.x is required for better performance.

3.1.1 Prerequisites

- A system running with Ubuntu 20.04
- A user account with sudo privileges
- Access to the command line

3.1.2 Installation

Python3 is pre-installed in Debian Linux versions such as Ubuntu 20.04 and others. To ensure that the version of python is recent, we will update the local package index.

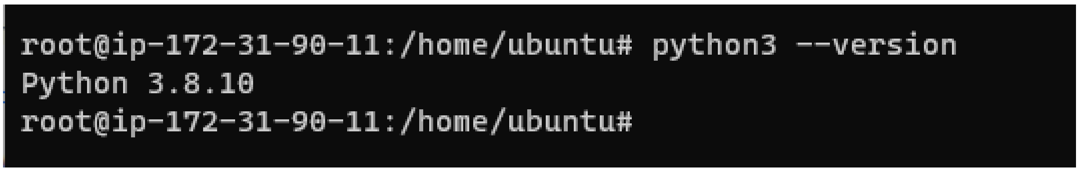
$ sudo apt update

Upgrading the packages will help in getting the latest version.

$ sudo apt -y upgrade

When the procedure is finished, we can use the following command to see what version of Python 3 is already installed on the system:

$ python3 --version



```
root@ip-172-31-90-11:/home/ubuntu# python3 --version
Python 3.8.10
root@ip-172-31-90-11:/home/ubuntu#
```

Figure 3 Python version

We can see the version we have is Python 3.8.10.

3.2 Honeypot

We use Dionaea as the honey pot. It intends to capture malware that makes use of the flows revealed by services provided through a network to eventually get a copy of the malware or virus. It offers several services to attract adversaries like SMB, HTTP, FTP, TFTP, VoIP, MSSQL, etc.

3.2.1 Prerequisites

- Ubuntu server 18.04 or 22.04
- Recommended to host on a public VPS

3.2.2 Installation

Dionaea needs to be compiled as it doesn't come in that way. We start the installation by downloading the source code from GitHub.

```
$ cd ~
$ git clone https://github.com/DinoTools/dionaea.git
$ cd dionaea
```

Ubuntu 22.04 doesn't come with the libemu-dev package. Hence install the package before the installing dependencies. If not the dependencies won't get installed completely.

```
$ sudo apt-get install -y libemu-dev
```

Install all the compiler's dependencies in the next step.

```
$ sudo apt-get install \
build-essential \
cmake \
check \
cython3 \
libcurl4-openssl-dev \
libemu-dev \
libev-dev \
libglib2.0-dev \
libloudmouth1-dev \
libnetfilter-queue-dev \
libnl-3-dev \
libpcap-dev \
libssl-dev \
libtool \
libudns-dev \
python3 \
python3-dev \
python3-bson \
python3-yaml \
python3-boto3 \
fonts-liberation
```

We will establish a build directory and use CMake to configure the build process after all the dependencies are in place.

```
$ sudo mkdir build
$ cd build
$ sudo cmake -DCMAKE_INSTALL_PREFIX:PATH=/opt/dionaea ..
```

To compile it now, we'll use make, and to install it on our present system, we'll use make install.

```
$ sudo make
$ sudo make install
```

Dionaea will be installed now under /opt/dionaea

3.2.3 Configuration

There are mainly 4 directories that need to be considered while configuring Dionaea under */opt/dionaea/etc/dionaea/*. They are.

- ihandlers-available
- ihandlers-enabled
- services-available
- services-enabled

ihandlers are used to handle the traffic when a copy of the malware is sent to the honeypot. While ihandler-enabled provides a series of symbolic links pointing to configuration files in the "ihandlers-available," ihandler-available refers to the many plugins we may activate for dionaea.

The services directory refers to the actual protocols mimicked by Dionaea. To make the honeypot more realistic, very few services have been exposed. The unwanted protocols are removed by deleting the symbolic links in the services-enabled folder or else by commenting out each line in the yaml file for each service.

```
$ cd /opt/dionaea/etc/dionaea/services-enabled
$ sudo rm blackhole.yaml epmap.yaml ftp.yaml memcache.yaml mirror.yaml mongo.yaml
mqtt.yaml mssql.yaml pptp.yaml sip.yaml tftp.yaml upnp.yaml printer.yaml
```

3.2.4 Configuring Dionaea as a service

To manage Dionaea and to make the process easier, using systemd, is made as a service in the background by creating a new file */etc/systemd/system*.

```
$ sudo nano /etc/systemd/system/dionaea.service
```

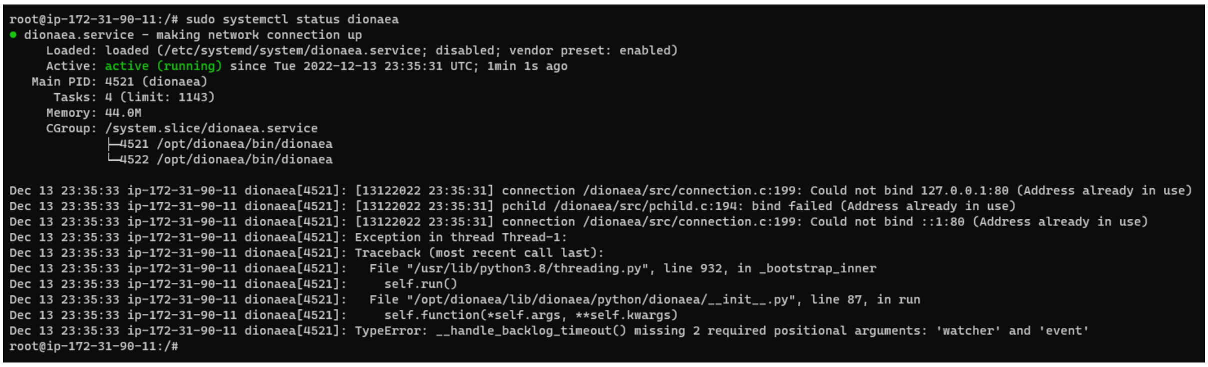
Paste the below details into the file and save.

```
[Unit]
```

Description = making network connection up
After = network.target
[Service]
ExecStart = /opt/dionaea/bin/dionaea
[Install]
WantedBy = multi-user.targetI

Now start Dionaea by using the systemctl command.

$ systemctl start Dionaea



```
root@ip-172-31-90-11:/# sudo systemctl status dionaea
● dionaea.service - making network connection up
     Loaded: loaded (/etc/systemd/system/dionaea.service; disabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-12-13 23:35:31 UTC; 1min 1s ago
   Main PID: 4521 (dionaea)
      Tasks: 4 (limit: 1143)
     Memory: 44.0M
     CGroup: /system.slice/dionaea.service
             ├─4521 /opt/dionaea/bin/dionaea
             └─4522 /opt/dionaea/bin/dionaea

Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: [13122022 23:35:31] connection /dionaea/src/connection.c:199: Could not bind 127.0.0.1:80 (Address already in use)
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: [13122022 23:35:31] pchild /dionaea/src/pchild.c:194: bind failed (Address already in use)
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: [13122022 23:35:31] connection /dionaea/src/connection.c:199: Could not bind ::1:80 (Address already in use)
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: Exception in thread Thread-1:
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: Traceback (most recent call last):
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]:   File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]:     self.run()
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]:   File "/opt/dionaea/lib/dionaea/python/dionaea/__init__.py", line 87, in run
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]:     self.function(*self.args, **self.kwargs)
Dec 13 23:35:33 ip-172-31-90-11 dionaea[4521]: TypeError: __handle_backlog_timeout() missing 2 required positional arguments: 'watcher' and 'event'
root@ip-172-31-90-11:/#
```

Figure 4 Dionaea status

3.2.5 Additional configuration

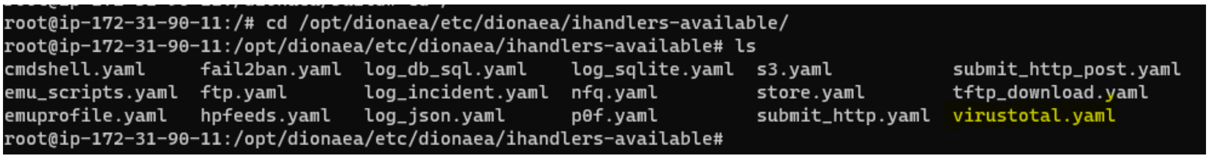
Automatically submitting captured binaries to Virus Total allows us to assist the community while also receiving an automated virus scan of the binaries captured. For that, we need a virus total account and API key provided by Virus Total. Create virustotal.yaml file inside ihandlers-available directory and update the API Key.

$ sudo nano /opt/dionaea/etc/dionaea/ihandlers-available/virustotal.yaml

apikey: "........."

The above ihandler can be enabled by creating a symbolic link

$ cd /opt/dionaea/etc/dionaea/ihandlers-available/
$ sudo ln -s ../ihandlers-available/virustotal.yaml ../ihandlers-enabled/virustotal.yaml



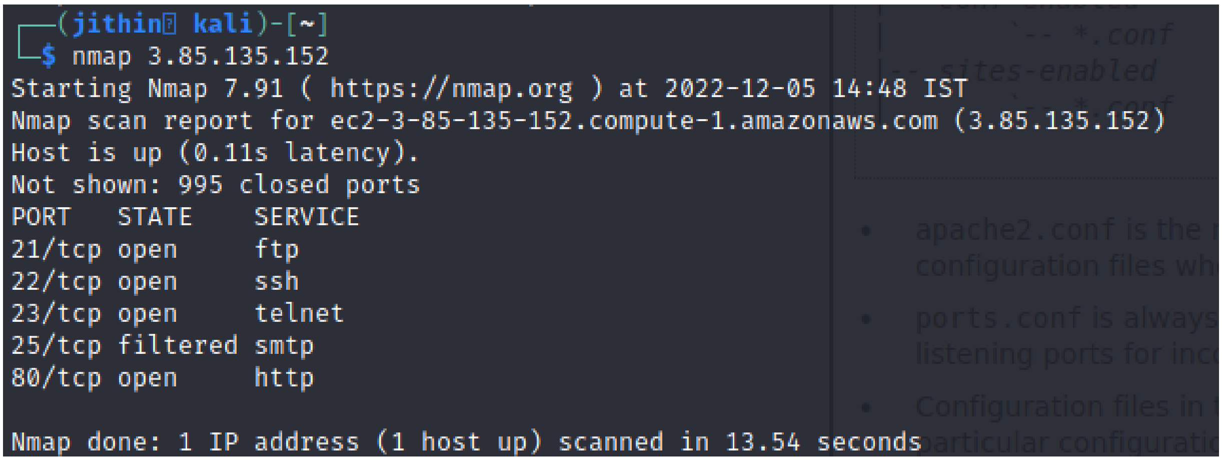
```
root@ip-172-31-90-11:/# cd /opt/dionaea/etc/dionaea/ihandlers-available/
root@ip-172-31-90-11:/opt/dionaea/etc/dionaea/ihandlers-available# ls
cmdshell.yaml     fail2ban.yaml  log_db_sql.yaml    log_sqlite.yaml  s3.yaml           submit_http_post.yaml
emu_scripts.yaml  ftp.yaml       log_incident.yaml  nfq.yaml         store.yaml        tftp_download.yaml
emuprofile.yaml   hpfeeds.yaml   log_json.yaml      p0f.yaml         submit_http.yaml  virustotal.yaml
root@ip-172-31-90-11:/opt/dionaea/etc/dionaea/ihandlers-available#
```

Figure 5 Additional configuration file for Virus Total

Restart Dionaea for better performance.

$ sudo systemctl restart Dionaea

Below are the services mimicked by Dionaea.



```
┌──(jithin㉿ kali)-[~]
└─$ nmap 3.85.135.152
Starting Nmap 7.91 ( https://nmap.org ) at 2022-12-05 14:48 IST
Nmap scan report for ec2-3-85-135-152.compute-1.amazonaws.com (3.85.135.152)
Host is up (0.11s latency).
Not shown: 995 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
23/tcp   open     telnet
25/tcp   filtered smtp
80/tcp   open     http

Nmap done: 1 IP address (1 host up) scanned in 13.54 seconds
```

Figure 6 Open ports/services mimicked by Dionaea honeypot

3.3 Custom IDS & Dashboard Installation

Custom IDS is a python program that segregates the captured packets based on signature-based and heuristic-based detection methods. The basic script has been taken from GitHub and developed by adding more features like the TOR browser concept, and DNS sinkhole, by updating and adding more entry files for suspicious, malicious, and malware-related packet capturing. The modified script is uploaded to GitHub and cloned from there.

Start the implementation by downloading the code.

$ git clone https://github.com/Jithinpj9/HDS.git

The system has been named HoneyDS by combining Honeypot and IDS. Hence created a directory honeyds and moved the files to it.

```
$ mv HDS honeyds
$ cd honeyds
$ bash deploy_server.sh
$ cd ..
$ cp -r  honeyds /tmp
$ cd /tmp/
$ cd honeyds/
$ bash deploy_server.sh
```

Now, pcapy must be installed for analyzing the captured packets.

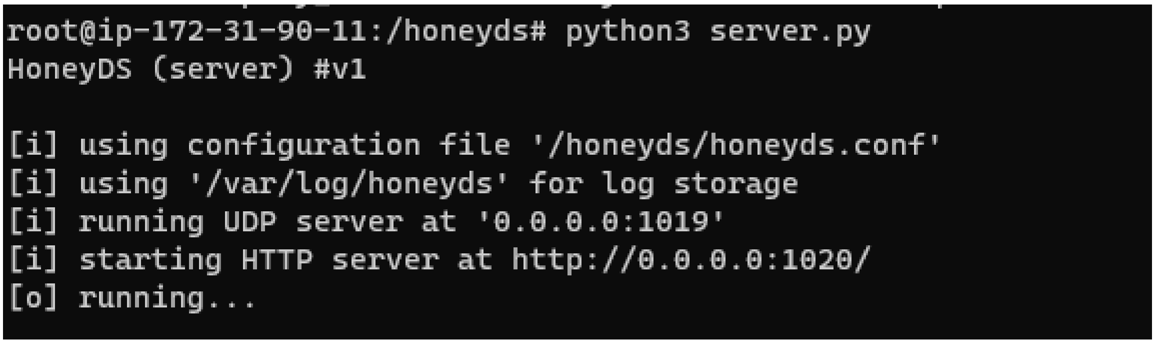
```
$ sudo apt install python3-pcapy
```

After that we need to deploy server.py (dashboard) and sensor.py (custom IDS)

```
$ bash deploy_server.sh
$ bash deploy_sensor.sh
```

Now custom IDS and the dashboard has been implemented. We need to turn on the capturing mode for IDS and the dashboard using the below commands.

$ python3 server.py
$ python3 sensor.py



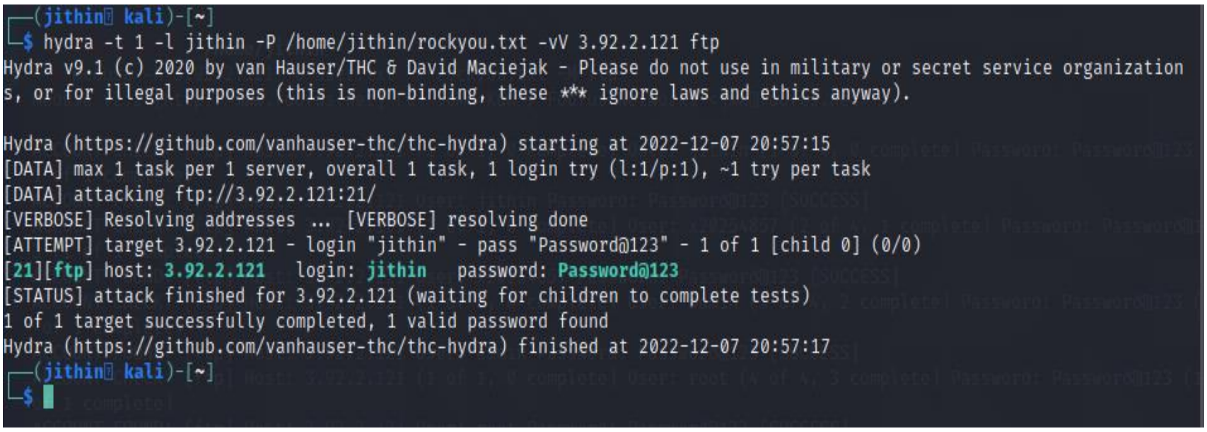
```
root@ip-172-31-90-11:/honeyds# python3 server.py
HoneyDS (server) #v1

[i] using configuration file '/honeyds/honeyds.conf'
[i] using '/var/log/honeyds' for log storage
[i] running UDP server at '0.0.0.0:1019'
[i] starting HTTP server at http://0.0.0.0:1020/
[o] running...
```

Figure 7 Custom IDS in capturing mode

Now both the sensor and dashboard are up and running.

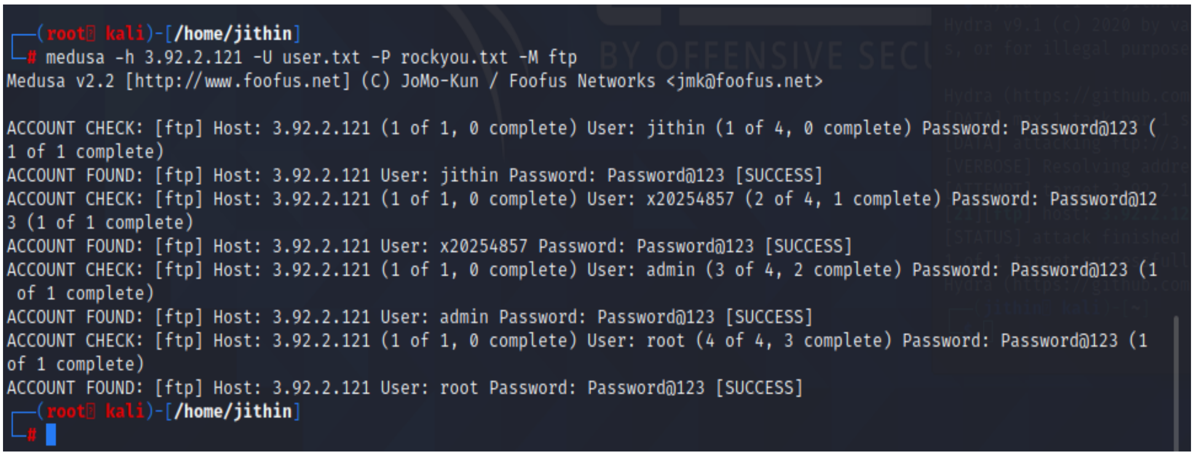
4 Attack Simulation and Packet Capture

The attacks have been simulated from the Kali machine running on a VirtualBox to test the efficiency of the implemented project. Several attacks and port scanning has been performed using Nmap scan, password attack has been performed using Hydra, Medusa, and Metasploit framework for probing the vulnerable services. Dirbuster is also used to brute force directories and file names through HTTP port.



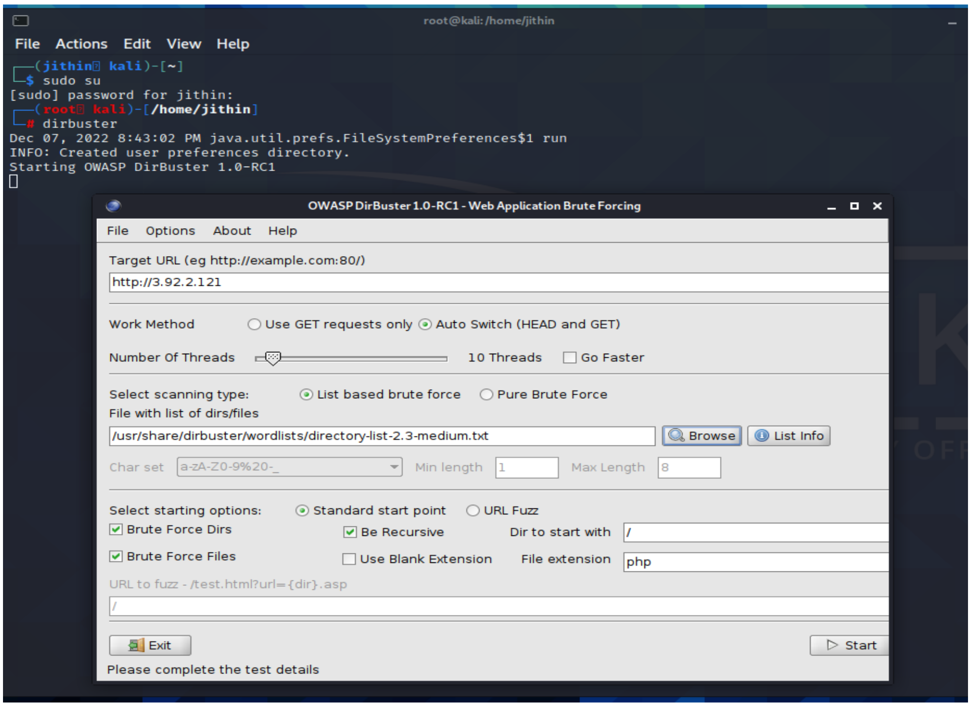
```
┌──(jithin㉿ kali)-[~]
└─$ hydra -t 1 -l jithin -P /home/jithin/rockyou.txt -vV 3.92.2.121 ftp
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization
s, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-12-07 20:57:15
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ftp://3.92.2.121:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 3.92.2.121 - login "jithin" - pass "Password@123" - 1 of 1 [child 0] (0/0)
[21][ftp] host: 3.92.2.121   login: jithin   password: Password@123
[STATUS] attack finished for 3.92.2.121 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-12-07 20:57:17
┌──(jithin㉿ kali)-[~]
└─$ ▋
```

Figure 8 Password brute force attack using hydra on ftp



```
┌──(root□ kali)-[/home/jithin]
└─# medusa -h 3.92.2.121 -U user.txt -P rockyou.txt -M ftp
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ftp] Host: 3.92.2.121 (1 of 1, 0 complete) User: jithin (1 of 4, 0 complete) Password: Password@123 (
1 of 1 complete)
ACCOUNT FOUND: [ftp] Host: 3.92.2.121 User: jithin Password: Password@123 [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 3.92.2.121 (1 of 1, 0 complete) User: x20254857 (2 of 4, 1 complete) Password: Password@12
3 (1 of 1 complete)
ACCOUNT FOUND: [ftp] Host: 3.92.2.121 User: x20254857 Password: Password@123 [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 3.92.2.121 (1 of 1, 0 complete) User: admin (3 of 4, 2 complete) Password: Password@123 (1
 of 1 complete)
ACCOUNT FOUND: [ftp] Host: 3.92.2.121 User: admin Password: Password@123 [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 3.92.2.121 (1 of 1, 0 complete) User: root (4 of 4, 3 complete) Password: Password@123 (1
of 1 complete)
ACCOUNT FOUND: [ftp] Host: 3.92.2.121 User: root Password: Password@123 [SUCCESS]
┌──(root□ kali)-[/home/jithin]
└─# 
```

Figure 99 Password brute forcing using medusa on ftp



root@kali: /home/jithin
File Actions Edit View Help

```
┌──(jithin□ kali)-[~]
└─$ sudo su
[sudo] password for jithin:
┌──(root□ kali)-[/home/jithin]
└─# dirbuster
Dec 07, 2022 8:43:02 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Starting OWASP DirBuster 1.0-RC1
```

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help

Target URL (eg http://example.com:80/): http://3.92.2.121

Work Method: ○ Use GET requests only ● Auto Switch (HEAD and GET)

Number Of Threads: 10 Threads □ Go Faster

Select scanning type: ● List based brute force ○ Pure Brute Force

File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: ● Standard start point ○ URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

☑ Brute Force Files □ Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp: /

Please complete the test details

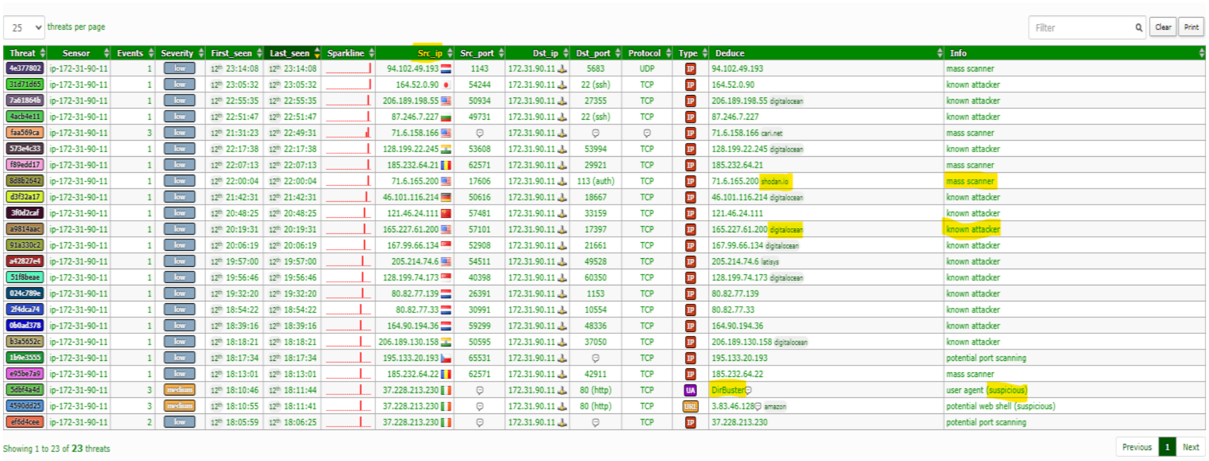
Figure 10 Directory brute forcing using Dirbuster

We can see the IDS has segregated the packets and we can see the details on the dashboard. Dashboad can be access by using the below link

http://public ip of the instance:1020

The default username and password used to access the dashboard is admin.

The dashboard has two windows that can be switched using the button on top named “Home” and “Normal”. The home window shows malicious, suspicious attack details and the normal window shows unknown traffic.

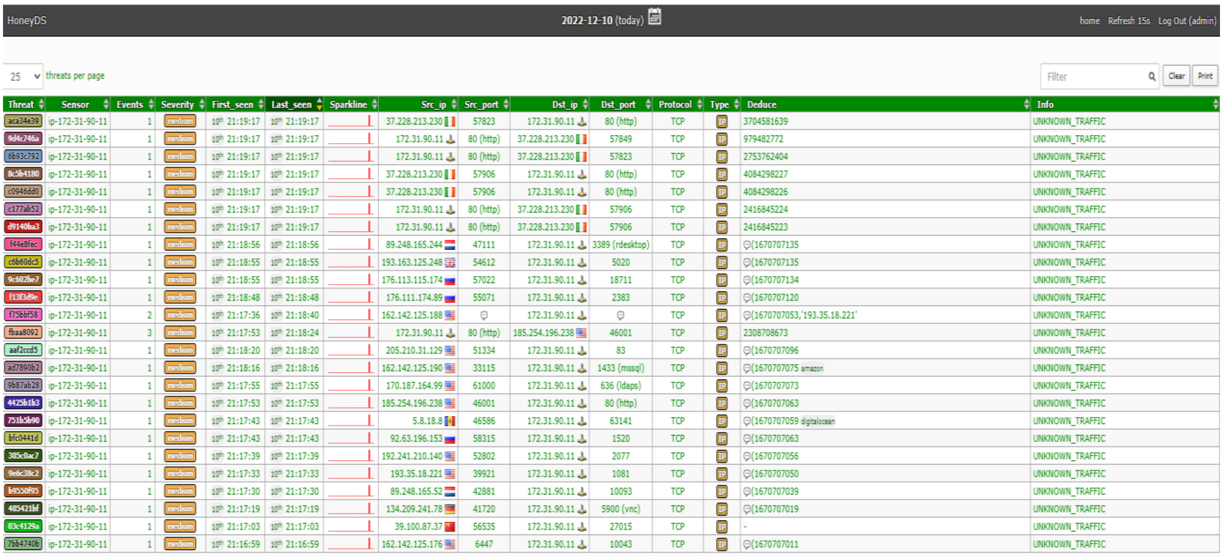


Threat	Sensor	Events	Severity	First_seen	Last_seen	Sparkline	Src_ip	Src_port	Dst_ip	Dst_port	Protocol	Type	Deduce	Info
4e377802	ip-172-31-90-11	1	low	12th 23:14:08	12th 23:14:08		94.102.49.193	1143	172.31.90.11	5683	UDP	IP	94.102.49.193	mass scanner
31d71d65	ip-172-31-90-11	1	low	12th 23:05:32	12th 23:05:32		164.52.0.90	54244	172.31.90.11	22 (ssh)	TCP	IP	164.52.0.90	known attacker
7a61864b	ip-172-31-90-11	1	low	12th 22:55:35	12th 22:55:35		206.189.198.55	50934	172.31.90.11	27355	TCP	IP	206.189.198.55 digitalocean	known attacker
4acb4e11	ip-172-31-90-11	1	low	12th 22:51:47	12th 22:51:47		87.246.7.227	49731	172.31.90.11	22 (ssh)	TCP	IP	87.246.7.227	known attacker
faa569ca	ip-172-31-90-11	3	low	12th 21:31:23	12th 22:49:31		71.6.158.166		172.31.90.11			IP	71.6.158.166 cari.net	mass scanner
573e4c33	ip-172-31-90-11	1	low	12th 22:17:38	12th 22:17:38		128.199.22.245	53608	172.31.90.11	53994	TCP	IP	128.199.22.245 digitalocean	known attacker
f89edd17	ip-172-31-90-11	1	low	12th 22:07:13	12th 22:07:13		185.232.64.21	62571	172.31.90.11	29921	TCP	IP	185.232.64.21	mass scanner
8d8b2642	ip-172-31-90-11	1	low	12th 22:00:04	12th 22:00:04		71.6.165.200	17606	172.31.90.11	113 (auth)	TCP	IP	71.6.165.200 shodan.io	mass scanner
d3f32a17	ip-172-31-90-11	1	low	12th 21:42:31	12th 21:42:31		46.101.116.214	50616	172.31.90.11	18667	TCP	IP	46.101.116.214 digitalocean	known attacker
3f0d2caf	ip-172-31-90-11	1	low	12th 20:48:25	12th 20:48:25		121.46.24.111	57481	172.31.90.11	33159	TCP	IP	121.46.24.111	known attacker
a9814aac	ip-172-31-90-11	1	low	12th 20:19:31	12th 20:19:31		165.227.61.200	57101	172.31.90.11	17397	TCP	IP	165.227.61.200 digitalocean	known attacker
91a330c2	ip-172-31-90-11	1	low	12th 20:06:19	12th 20:06:19		167.99.66.134	52908	172.31.90.11	21661	TCP	IP	167.99.66.134 digitalocean	known attacker
a42827e4	ip-172-31-90-11	1	low	12th 19:57:00	12th 19:57:00		205.214.74.6	54511	172.31.90.11	49528	TCP	IP	205.214.74.6 latisys	known attacker
51f8beae	ip-172-31-90-11	1	low	12th 19:56:46	12th 19:56:46		128.199.74.173	40398	172.31.90.11	60350	TCP	IP	128.199.74.173 digitalocean	known attacker
024c789e	ip-172-31-90-11	1	low	12th 19:32:20	12th 19:32:20		80.82.77.139	26391	172.31.90.11	1153	TCP	IP	80.82.77.139	known attacker
2f4dca74	ip-172-31-90-11	1	low	12th 18:54:22	12th 18:54:22		80.82.77.33	30991	172.31.90.11	10554	TCP	IP	80.82.77.33	known attacker
0b0ad378	ip-172-31-90-11	1	low	12th 18:39:16	12th 18:39:16		164.90.194.36	59299	172.31.90.11	48336	TCP	IP	164.90.194.36	known attacker
b3a5652c	ip-172-31-90-11	1	low	12th 18:18:21	12th 18:18:21		206.189.130.158	50595	172.31.90.11	37050	TCP	IP	206.189.130.158 digitalocean	known attacker
1b9e3555	ip-172-31-90-11	1	low	12th 18:17:34	12th 18:17:34		195.133.20.193	65531	172.31.90.11		TCP	IP	195.133.20.193	potential port scanning
e95be7a9	ip-172-31-90-11	1	low	12th 18:13:01	12th 18:13:01		185.232.64.22	62571	172.31.90.11	42911	TCP	IP	185.232.64.22	mass scanner
5dbf4a4d	ip-172-31-90-11	3	medium	12th 18:10:46	12th 18:11:44		37.228.213.230		172.31.90.11	80 (http)	TCP	UA	DirBuster	user agent (suspicious)
4590dd25	ip-172-31-90-11	3	medium	12th 18:10:55	12th 18:11:41		37.228.213.230		172.31.90.11	80 (http)	TCP	URL	3.83.46.128 amazon	potential web shell (suspicious)
ef6d4cee	ip-172-31-90-11	2	low	12th 18:05:59	12th 18:06:25		37.228.213.230		172.31.90.11		TCP	IP	37.228.213.230	potential port scanning

Showing 1 to 23 of 23 threats

Figure 10 Malicious attacks details on the dashboard

The above diagram shows the Home window where the attacks have been captured. As the project was deployed on public IP, the system was able to capture packets sent by known attackers, and the mass scanning was done using Shodan, and other scanners hosted on a cloud environment. Also, the geo-location of the source IP is represented using the respective national flag.



Threat	Sensor	Events	Severity	First_seen	Last_seen	Sparkline	Src_ip	Src_port	Dst_ip	Dst_port	Protocol	Type	Deduce	Info
aca34e39	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		37.228.213.230	57823	172.31.90.11	80 (http)	TCP	IP	3704581639	UNKNOWN_TRAFFIC
9d4c746a	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		172.31.90.11	80 (http)	37.228.213.230	57849	TCP	IP	979482772	UNKNOWN_TRAFFIC
6b93c792	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		172.31.90.11	80 (http)	37.228.213.230	57823	TCP	IP	2753762404	UNKNOWN_TRAFFIC
8c5b4180	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		37.228.213.230	57906	172.31.90.11	80 (http)	TCP	IP	4084298227	UNKNOWN_TRAFFIC
c0946dd0	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		37.228.213.230	57906	172.31.90.11	80 (http)	TCP	IP	4084298226	UNKNOWN_TRAFFIC
c177ab52	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		172.31.90.11	80 (http)	37.228.213.230	57906	TCP	IP	2416845224	UNKNOWN_TRAFFIC
d9140ba3	ip-172-31-90-11	1	medium	10th 21:19:17	10th 21:19:17		172.31.90.11	80 (http)	37.228.213.230	57906	TCP	IP	2416845223	UNKNOWN_TRAFFIC
f44e8fec	ip-172-31-90-11	1	medium	10th 21:18:56	10th 21:18:56		89.248.165.244	47111	172.31.90.11	3389 (rdesktop)	TCP	IP	(1670707135	UNKNOWN_TRAFFIC
c6b60dc5	ip-172-31-90-11	1	medium	10th 21:18:55	10th 21:18:55		193.163.125.248	54612	172.31.90.11	5020	TCP	IP	(1670707135	UNKNOWN_TRAFFIC
9c602be7	ip-172-31-90-11	1	medium	10th 21:18:55	10th 21:18:55		176.113.115.174	57022	172.31.90.11	18711	TCP	IP	(1670707134	UNKNOWN_TRAFFIC
f13f3d9e	ip-172-31-90-11	1	medium	10th 21:18:48	10th 21:18:48		176.111.174.89	55071	172.31.90.11	2383	TCP	IP	(1670707120	UNKNOWN_TRAFFIC
f75bbf58	ip-172-31-90-11	2	medium	10th 21:17:36	10th 21:18:40		162.142.125.188		172.31.90.11		TCP	IP	(1670707053,'193.35.18.221'	UNKNOWN_TRAFFIC
fbaa8092	ip-172-31-90-11	3	medium	10th 21:17:53	10th 21:18:24		172.31.90.11	80 (http)	185.254.196.238	46001	TCP	IP	2308708673	UNKNOWN_TRAFFIC
aaf2ccd5	ip-172-31-90-11	1	medium	10th 21:18:20	10th 21:18:20		205.210.31.129	51334	172.31.90.11	83	TCP	IP	(1670707096	UNKNOWN_TRAFFIC
ad7890b2	ip-172-31-90-11	1	medium	10th 21:18:16	10th 21:18:16		162.142.125.190	33115	172.31.90.11	1433 (mssql)	TCP	IP	(1670707075 amazon	UNKNOWN_TRAFFIC
9b87ab28	ip-172-31-90-11	1	medium	10th 21:17:55	10th 21:17:55		170.187.164.99	61000	172.31.90.11	636 (ldaps)	TCP	IP	(1670707073	UNKNOWN_TRAFFIC
4425b1b3	ip-172-31-90-11	1	medium	10th 21:17:53	10th 21:17:53		185.254.196.238	46001	172.31.90.11	80 (http)	TCP	IP	(1670707063	UNKNOWN_TRAFFIC
751b5b90	ip-172-31-90-11	1	medium	10th 21:17:43	10th 21:17:43		5.8.18.8	46586	172.31.90.11	63141	TCP	IP	(1670707059 digitalocean	UNKNOWN_TRAFFIC
bfc0441d	ip-172-31-90-11	1	medium	10th 21:17:43	10th 21:17:43		92.63.196.153	58315	172.31.90.11	1520	TCP	IP	(1670707063	UNKNOWN_TRAFFIC
305c0ac7	ip-172-31-90-11	1	medium	10th 21:17:39	10th 21:17:39		192.241.210.140	52802	172.31.90.11	2077	TCP	IP	(1670707056	UNKNOWN_TRAFFIC
9e6c38c2	ip-172-31-90-11	1	medium	10th 21:17:33	10th 21:17:33		193.35.18.221	39921	172.31.90.11	1081	TCP	IP	(1670707050	UNKNOWN_TRAFFIC
b9550f95	ip-172-31-90-11	1	medium	10th 21:17:30	10th 21:17:30		89.248.165.52	42881	172.31.90.11	10093	TCP	IP	(1670707039	UNKNOWN_TRAFFIC
405421bf	ip-172-31-90-11	1	medium	10th 21:17:19	10th 21:17:19		134.209.241.78	41720	172.31.90.11	5900 (vnc)	TCP	IP	(1670707019	UNKNOWN_TRAFFIC
03c4129a	ip-172-31-90-11	1	medium	10th 21:17:03	10th 21:17:03		39.100.87.37	56535	172.31.90.11	27015	TCP	IP	-	UNKNOWN_TRAFFIC
7bb4740b	ip-172-31-90-11	1	medium	10th 21:16:59	10th 21:16:59		162.142.125.176	6447	172.31.90.11	10043	TCP	IP	(1670707011	UNKNOWN_TRAFFIC

Figure 11 Unknown attack detail

References

admin (2019) ‘Dionaea - Setting up a Honeypot environment (Part 2)’, *blogg.kroland.no*, 14 October. Available at: https://kroland.no/2019/10/14/dionaea-setting-up-a-honeypot-environment-part-2/ (Accessed: 15 November 2022).

Andy Smith’s Blog - Dionaea Honeypot on EC2 in 20 minutes (no date). Available at: https://andrewmichaelsmith.com/2012/03/dionaea-honeypot-on-ec2-in-20-minutes/ (Accessed: 10 November 2022).

Installation — dionaea 0.11.0 documentation (no date). Available at:

https://dionaea.readthedocs.io/en/latest/installation.html (Accessed: 14 November 2022).

Editor, D.B. (2014) *Dionaea – A Malware Capturing Honeypot*, *Division Zero (Div0)*. Available at: https://www.div0.sg/post/dionaea (Accessed: 12 November 2022).

Jithinpj9/HDS (no date). Available at: https://github.com/Jithinpj9/HDS (Accessed: 24 November 2022).

sharathc213 (2021) ‘HoneyDS’. Available at: https://github.com/sharathc213/HoneyDs (Accessed: 20 October 2022).