Université Euro-Méditerranéenne Fès

Ecole d'Ingénierie Digitale et d'Intelligence Artificielle
(EIDIA)

Simulating Cyber Attacks: A Comprehensive Honeypot-Based Approach to Network Security

4th year Cybersecurity

2024/2025

Supervised by :
Mr. Airaj Mohammed

Prepared by :
Fatima BOUYARMANE
Wissal BOUTAYEB

Abstract

As cyber threats continue to evolve, organizations require innovative solutions to detect, analyze, and mitigate malicious activities targeting their networks. This project combines several advanced security technologies, including honeypots, security confinement tools, and data analysis frameworks, to create a robust security monitoring and defense system. The project leverages multiple honeypot solutions—Dionaea, Cowrie, and Honeyd—to simulate a variety of network vulnerabilities and attract malicious actors, enabling the collection of real-world attack data. These honeypots provide valuable insights into attack methodologies, such as exploits, malware delivery, and command-and-control operations, which can be used for proactive defense strategies.

In addition to honeypots, the project incorporates security confinement technologies—Seccomp and AppArmor—to enhance the system's ability to isolate and protect critical processes. Seccomp is used to limit the system calls that can be executed by processes, reducing the risk of exploitation, while AppArmor applies mandatory access control (MAC) policies to restrict the actions of programs. Together, these technologies create a layered security approach that minimizes the attack surface and prevents potential exploits.
The collected data from the honeypots is analyzed using the ELK stack (Elasticsearch, Logstash, and Kibana). Logstash processes and transforms raw attack data, Elasticsearch indexes and stores the data, while Kibana provides visualization tools to detect attack patterns and monitor the health of the network. This integration enables real-time monitoring, alerting, and in-depth analysis of security events.

This project aims to develop an advanced, multi-faceted security system that not only identifies threats but also empowers security teams to respond proactively. By combining deception technologies (honeypots) with confinement techniques and powerful analytics tools, the project seeks to provide a comprehensive defense against modern cyberattacks, ultimately strengthening an organization's cybersecurity posture and reducing the likelihood of successful breaches.

Table of contents

I. Introduction

1.1 Context and Objectives of the Project

In the evolving landscape of cybersecurity threats, organizations must enhance their defenses and anticipate attacks. Security Information and Event Management (SIEM) systems play a critical role by collecting, analyzing, and correlating real-time data to detect intrusions. To go beyond traditional detection methods, honeypots offer a proactive approach by attracting and studying attackers' activities.

The objectives of this project are:

- **Deploying and configuring honeypots** (Dionaea, Cowrie, and Honeyd) to simulate vulnerable systems and observe attackers' tactics, techniques, and procedures (TTPs).
- **Isolating attackers from the real network** to prevent any impact on critical systems while allowing interaction with seemingly vulnerable resources.
- **Enhancing honeypot security** using advanced concepts like Seccomp, AppArmor, and SELinux to limit potential malicious actions.
- **Visualizing attack data** by integrating **logs** into the ELK stack (Elasticsearch, Logstash, Kibana) for in-depth and intuitive analysis.

1.2 Overview of Tools Used

To achieve the objectives, several tools were selected for their relevance and robust features:

Honeypots:

- Dionaea: Designed to capture exploits with a focus on vulnerabilities in common network services. It collects data on attack attempts and malicious payloads.
- Cowrie: An interactive SSH/Telnet honeypot that simulates a vulnerable shell, capturing attackers' interactions and commands.
- Honeyd: A flexible honeypot capable of emulating multiple operating systems and services to attract various types of attacks.

Security Hardening:

- Seccomp (Secure Computing Mode): A Linux security mechanism that restricts the system calls an application can make, reducing the attack surface.
- AppArmor (Application Armor): An access control system that applies predefined profiles to limit application actions.

ELK Stack:

- Elasticsearch: A search and analytics engine for indexing and querying honeypot logs.
- Logstash: A tool for collecting, transforming, and sending logs to Elasticsearch.
- Kibana: A visualization platform that provides insights into attack data through interactive dashboards.
- By combining these tools, the project provides a comprehensive environment to monitor, analyze, and mitigate security threats effectively.

II. Overview of Honeypots

Honeypots are specialized tools designed to mimic vulnerable systems and attract attackers, enabling organizations to study their behavior and gather valuable intelligence. Below is a detailed exploration of the three honeypots used in this project.

2.1 Dionaea: Features and Role

Dionaea is a low-interaction honeypot designed to capture malware and exploits. It primarily focuses on vulnerabilities in network services to attract and analyze malicious activity.

Features:

- Protocol Support: Dionaea supports various protocols such as SMB, FTP, HTTP, HTTPS, TFTP, and more, allowing it to emulate multiple types of services.
- Payload Capture: It captures malware payloads and stores them for further analysis, making it an invaluable tool for studying the behavior of malicious software.
- Integration with Analysis Tools: Logs can be forwarded to other systems, such as Elasticsearch, for advanced analysis and visualization.
- SQLite Database: All interaction data is logged into an SQLite database, providing easy access for querying and report generation.
- IPv6 Support: It is compatible with IPv6, enabling it to attract attacks from modern network configurations.

Role in the Project:

Capturing Exploits: Dionaea is used to detect exploitation attempts on network services, helping us study the methods attackers use to gain unauthorized access.
Payload Analysis: By collecting malware samples, it contributes to understanding the types of threats targeting the system and their potential impact.

2.2 Cowrie: Features and Role

Cowrie is an interactive SSH and Telnet honeypot designed to simulate a vulnerable command-line environment. It enables researchers to observe attackers' commands and methods in a controlled environment.

Features:

- Session Logging: Records all attacker interactions, including commands, files downloaded, and even keylogging of passwords entered.
- File Emulation: Simulates a file system with fake data to make the environment appear realistic.
- Download Traps: Allows attackers to upload or download files, capturing these files for forensic analysis.
- Command Emulation: Supports several Unix shell commands, giving attackers the illusion of interacting with a real system.
- Credential Management: Logs all attempted SSH/Telnet credentials, providing insights into brute-force techniques.
- SSH Proxy Mode: Can act as a middleman to observe interactions with a legitimate server for advanced deception.

Role in the Project:

Brute Force Analysis: Cowrie is used to collect information about password brute-forcing attempts.

Behavior Observation: Allows detailed logging of the commands and actions taken by attackers once they gain access, providing a deeper understanding of their techniques and objectives.

2.3 Honeyd: Features and Role

Honeyd is a versatile honeypot capable of emulating a wide range of operating systems and services. It acts as a decoy to attract attackers by simulating realistic network environments.

Features:

- Network Simulation: Can emulate multiple virtual hosts with distinct IP addresses and configurations, creating a diverse attack surface.
- Service Emulation: Mimics various services such as web servers, mail servers, or custom protocols to attract a wide range of attacks.
- Customizable Fingerprints: Supports configuration of different operating system fingerprints, such as Windows, Linux, or BSD, to deceive attackers.
- Decoy Traffic: Generates fake network traffic to make the environment more convincing and attract scanning tools like Nmap.
- Lightweight: Minimal resource requirements make it suitable for deployment in various environments.
- Extensive Logging: Captures network interactions, providing detailed logs for further analysis.

Role in the Project:

Diversity of Attacks: Honeyd is deployed to attract attacks targeting different services and operating systems.

Network Decoy: It simulates multiple systems within a single deployment, diverting attackers from real systems and studying their reconnaissance techniques.

Attack Simulation: Provides a flexible platform for recreating specific attack scenarios to analyze vulnerabilities and response mechanisms.

III. Installation and configuration of Honeypots

This section outlines the steps taken to install the honeypot tools used in this project. It covers the installation process for Dionaea, Cowrie, and Honeyd, along with the specific configurations required for each honeypot to simulate different attack scenarios.

1.1. Dionaea Installation and Configuration :

Dionaea : Primarily designed to capture and emulate vulnerable services for malware collection, such as worms, viruses, and other types of malicious software. Dionaea specializes in capture malware, including exploits targeting vulnerabilities in services like SMB, HTTP, and FTP. It is often used to monitor exploit attempts and collect malware samples.

Fisrt, we installed all needed dependencies to ensure Dionaea works :

We cloned the repository of Dionaea from github :

We created the directory of build in the Dionaea directory, then we generated the configuration files with the **cmake** command.

```
wissal@ubuntu:~/dionaea$ mkdir build
wissal@ubuntu:~/dionaea$ cd build
wissal@ubuntu:~/dionaea/build$ cmake -DCMAKE_INSTALL_PREFIX:PATH=/opt/dionaea ..
-- The C compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Found Git: /usr/bin/git (found version "2.17.1")
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'glib-2.0>=2.30'
--   Found glib-2.0, version 2.56.4
-- Checking for module 'gmodule-2.0>=2.30'
--   Found gmodule-2.0, version 2.56.4
-- Checking for module 'libemu'
--   Found libemu, version 0.2.0
-- Checking for module 'libnetfilter_queue'
--   Found libnetfilter_queue, version 1.0.2
-- Checking for module 'libnl-3.0'
--   Found libnl-3.0, version 3.2.29
-- Checking for module 'libnl-route-3.0'
--   No package 'libnl-route-3.0' found
-- Found PythonInterp: /usr/bin/python3.6 (found version "3.6.9")
-- Found PythonLibs: /usr/lib/x86_64-linux-gnu/libpython3.6m.so (found suitable version "3.6.9", minimum required is "3.6.9")
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version "1.1.1")
-- Found UDNS: /usr/lib/x86_64-linux-gnu/libudns.so (found version "0.4")
-- Found EV: /usr/lib/x86_64-linux-gnu/libev.so (found version "4.22")
-- Found CURL: /usr/lib/x86_64-linux-gnu/libcurl.so (found version "7.58.0")
-- Found PCAP: /usr/lib/x86_64-linux-gnu/libpcap.so (found version "2.4")
example/form.html.j2;nginx/autoindex.html.j2;nginx/error.html.j2
-- Configuring done
-- Generating done
-- Build files have been written to: /home/wissal/dionaea/build
```

We used the **make** command to compile Dionaea :

```
wissal@ubuntu:~/dionaea/build$ make
Scanning dependencies of target dionaea
[  2%] Building C object src/CMakeFiles/dionaea.dir/dionaea.c.o
[  5%] Building C object src/CMakeFiles/dionaea.dir/bistream.c.o
[  8%] Building C object src/CMakeFiles/dionaea.dir/connection.c.o
[ 11%] Building C object src/CMakeFiles/dionaea.dir/connection_dtls.c.o
[ 14%] Building C object src/CMakeFiles/dionaea.dir/connection_tcp.c.o
[ 17%] Building C object src/CMakeFiles/dionaea.dir/connection_tls.c.o
[ 20%] Building C object src/CMakeFiles/dionaea.dir/connection_udp.c.o
[ 22%] Building C object src/CMakeFiles/dionaea.dir/dns.c.o
[ 25%] Building C object src/CMakeFiles/dionaea.dir/incident.c.o
[ 28%] Building C object src/CMakeFiles/dionaea.dir/log.c.o
[ 31%] Building C object src/CMakeFiles/dionaea.dir/modules.c.o
[ 34%] Building C object src/CMakeFiles/dionaea.dir/node_info.c.o
[ 37%] Building C object src/CMakeFiles/dionaea.dir/pchild.c.o
[ 40%] Building C object src/CMakeFiles/dionaea.dir/processor.c.o
[ 42%] Building C object src/CMakeFiles/dionaea.dir/refcount.c.o
[ 45%] Building C object src/CMakeFiles/dionaea.dir/signals.c.o
[ 48%] Building C object src/CMakeFiles/dionaea.dir/ssl.c.o
[ 51%] Building C object src/CMakeFiles/dionaea.dir/threads.c.o
[ 54%] Building C object src/CMakeFiles/dionaea.dir/util.c.o
[ 57%] Linking C executable dionaea
[ 57%] Built target dionaea
Scanning dependencies of target module_python
[ 60%] Building C object modules/python/CMakeFiles/module_python.dir/module.c.o
[ 62%] Linking C shared module python.so
[ 62%] Built target module_python
Scanning dependencies of target python_package
[ 65%] Generating build/timestamp
running build
running build_py
creating build
creating build/lib.linux-x86_64-3.6
creating build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/logsql.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/mirror.py -> build/lib.linux-x86_64-3.6/dionaea
running build_ext
cythoning /home/wissal/dionaea/modules/python/binding.pyx to /home/wissal/dionaea/modules/python/binding.c
building 'dionaea.core' extension
creating build/temp.linux-x86_64-3.6
creating build/temp.linux-x86_64-3.6/home
creating build/temp.linux-x86_64-3.6/home/wissal
creating build/temp.linux-x86_64-3.6/home/wissal/dionaea
creating build/temp.linux-x86_64-3.6/home/wissal/dionaea/modules
creating build/temp.linux-x86_64-3.6/home/wissal/dionaea/modules/python
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fPIC -D_GNU_SOURCE -UNDEBUG -I/home/wissal/dionaea/modules/python/../../include -I/home/wissal/dionaea/modules/python/../../ -I/usr/lib/x86_64-linux-gnu/glib-2.0/include -I/usr/include/glib-2.0 -I/usr/include/python3.6m -c /home/wissal/dionaea/modules/python/binding.c -o build/temp.linux-x86_64-3.6/home/wissal/dionaea/modules/python/binding.o -pthread
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-Bsymbolic-functions -Wl,-z,relro -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 build/temp.linux-x86_64-3.6/home/wissal/dionaea/modules/python/binding.o -lgmodule-2.0 -lglib-2.0 -o build/lib.linux-x86_64-3.6/dionaea/core.cpython-36m-x86_64-linux-gnu.so -pthread -Wl,--export-dynamic
[ 65%] Built target python_package
Scanning dependencies of target module_curl
[ 68%] Building C object modules/curl/CMakeFiles/module_curl.dir/module.c.o
[ 71%] Linking C shared module curl.so
[ 71%] Built target module_curl
Scanning dependencies of target module_emu
[ 74%] Building C object modules/emu/CMakeFiles/module_emu.dir/module.c.o
[ 77%] Building C object modules/emu/CMakeFiles/module_emu.dir/detect.c.o
[ 80%] Building C object modules/emu/CMakeFiles/module_emu.dir/emulate.c.o
[ 82%] Building C object modules/emu/CMakeFiles/module_emu.dir/profile.c.o
[ 85%] Building C object modules/emu/CMakeFiles/module_emu.dir/hooks.c.o
[ 88%] Linking C shared module emu.so
[ 88%] Built target module_emu
Scanning dependencies of target module_nfq
[ 91%] Building C object modules/nfq/CMakeFiles/module_nfq.dir/nfq.c.o
[ 94%] Linking C shared module nfq.so
[ 94%] Built target module_nfq
Scanning dependencies of target module_pcap
[ 97%] Building C object modules/pcap/CMakeFiles/module_pcap.dir/pcap.c.o
[100%] Linking C shared module pcap.so
[100%] Built target module_pcap
wissal@ubuntu:~/dionaea/build$
```

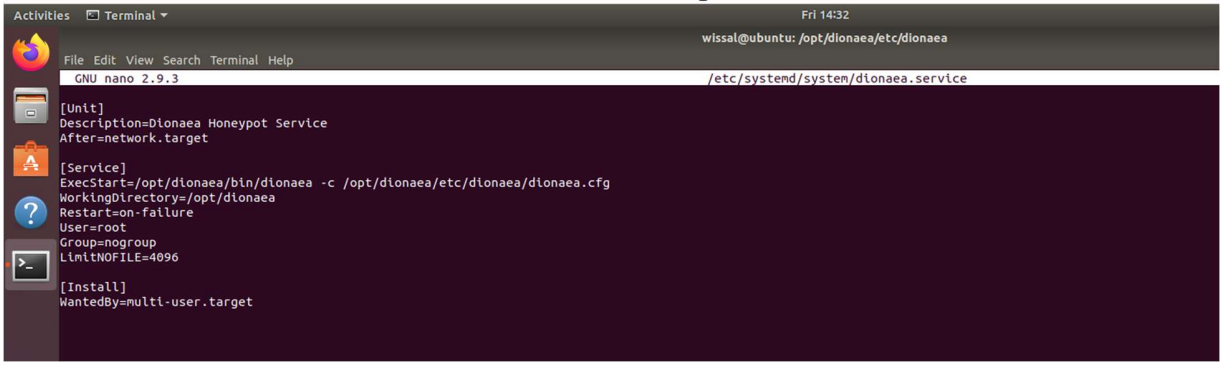
We used the command : **sudo make install**, for the installation of Dionaea :

```
wissal@ubuntu:~/dionaea/build$ sudo make install
[ 57%] Built target dionaea
[ 62%] Built target module_python
[ 65%] Built target python_package
[ 71%] Built target module_curl
[ 88%] Built target module_emu
[ 94%] Built target module_nfq
[100%] Built target module_pcap
Install the project...
-- Install configuration: "Debug"
-- Installing: /opt/dionaea/share/doc/dionaea/CHANGELOG.rst
-- Installing: /opt/dionaea/share/doc/dionaea/CONTRIBUTING.rst
-- Installing: /opt/dionaea/share/doc/dionaea/LICENSE
-- Installing: /opt/dionaea/share/doc/dionaea/README.md
-- Installing: /opt/dionaea/share/doc/dionaea/LICENSE.openssl
running install
running build
running build_py
creating build
creating build/lib.linux-x86_64-3.6
creating build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/logsql.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/mirror.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/blackhole.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/ihandlers.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/ftp.py -> build/lib.linux-x86_64-3.6/dionaea
copying /home/wissal/dionaea/modules/python/dionaea/cmd.py -> build/lib.linux-x86_64-3.6/dionaea
-- Installing: /opt/dionaea/etc/dionaea/ihandlers-available/submit_http.yaml
-- Installing: /opt/dionaea/etc/dionaea/ihandlers-available/tftp_download.yaml
-- Installing: /opt/dionaea/etc/dionaea/ihandlers-available/virustotal.yaml
-- Installing: /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: cmdshell.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: emuprofile.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: ftp.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: log_sqlite.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: store.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Enabling Service: tftp_download.yaml in /opt/dionaea/etc/dionaea/ihandlers-enabled
-- Installing: /opt/dionaea/var/lib/dionaea/fail2ban
-- Installing: /opt/dionaea/var/lib/dionaea/ftp/root
-- Installing: /opt/dionaea/var/lib/dionaea/http/root
-- Installing: /opt/dionaea/var/lib/dionaea/http/template
-- Installing: /opt/dionaea/var/lib/dionaea/http/template/example/form.html.j2
-- Installing: /opt/dionaea/var/lib/dionaea/http/template/nginx/autoindex.html.j2
-- Installing: /opt/dionaea/var/lib/dionaea/http/template/nginx/error.html.j2
-- Installing: /opt/dionaea/var/lib/dionaea/printer/root
-- Installing: /opt/dionaea/var/lib/dionaea/printer/root/0
-- Installing: /opt/dionaea/var/lib/dionaea/sip/
-- Installing: /opt/dionaea/var/lib/dionaea/sip/rtp
-- Installing: /opt/dionaea/var/lib/dionaea/tftp/root
-- Installing: /opt/dionaea/var/lib/dionaea/upnp/root
-- Installing: /opt/dionaea/lib/dionaea/curl.so
-- Installing: /opt/dionaea/lib/dionaea/emu.so
-- Installing: /opt/dionaea/lib/dionaea/nfq.so
-- Installing: /opt/dionaea/lib/dionaea/pcap.so
-- Installing: /opt/dionaea/bin/dionaea
-- Installing: /opt/dionaea/etc/dionaea/dionaea.cfg
-- Up-to-date: /opt/dionaea/var/lib/dionaea
-- Installing: /opt/dionaea/var/lib/dionaea/binaries
-- Installing: /opt/dionaea/var/lib/dionaea/bistreams
-- Installing: /opt/dionaea/var/log/dionaea
wissal@ubuntu:~/dionaea/build$
```

The creation of necessary directories for dionaea :

```
dionaea.cfg  ihandlers-available  ihandlers-enabled  services-available  services-enabled
wissal@ubuntu:/opt/dionaea/etc/dionaea$ cd services-enabled
wissal@ubuntu:/opt/dionaea/etc/dionaea/services-enabled$ ls
blackhole.yaml  epmap.yaml  ftp.yaml  http.yaml  memcache.yaml  mirror.yaml  mongo.yaml  mqtt.yaml  mssql.yaml  mysql.yaml  pptp.yaml  printer.yaml  sip.yaml  smb.yaml  tftp.yaml  upnp.yaml
wissal@ubuntu:/opt/dionaea/etc/dionaea/services-enabled$ cd ..
wissal@ubuntu:/opt/dionaea/etc/dionaea$
wissal@ubuntu:/opt/dionaea/etc/dionaea$
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo mkdir -p /opt/dionaea/var/log/dionaea
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo mkdir -p /opt/dionaea/var/lib/dionaea/binaries
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo mkdir -p /opt/dionaea/var/lib/dionaea/bistreams
```

The creation of the Dionaea service file, it contains the configuration to run Dionaea as a service :



This file defined the path to the main configuration file of Dionaea, the working directory, and the user and the group that has the priveliges to run this service.

We verified the status of Dionaea service, ensuring that it is active and running :

```
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo nano sudo nano /etc/systemd/system/dionaea.service
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo nano /etc/systemd/system/dionaea.service
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo nano /etc/systemd/system/dionaea.service
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo systemctl daemon-reload
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo systemctl restart dionaea
wissal@ubuntu:/opt/dionaea/etc/dionaea$ sudo systemctl status dionaea
● dionaea.service - Dionaea Honeypot Service
   Loaded: loaded (/etc/systemd/system/dionaea.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-12-27 14:30:21 PST; 12s ago
 Main PID: 15539 (dionaea)
    Tasks: 4 (limit: 4620)
   CGroup: /system.slice/dionaea.service
           ├─15539 /opt/dionaea/bin/dionaea -c /opt/dionaea/etc/dionaea/dionaea.cfg
           └─15542 /opt/dionaea/bin/dionaea -c /opt/dionaea/etc/dionaea/dionaea.cfg

Dec 27 14:30:21 ubuntu systemd[1]: Started Dionaea Honeypot Service.
wissal@ubuntu:/opt/dionaea/etc/dionaea$
```

Then we configured Dionaea file including the network interface, the paths to logs and errors, and services and modules such as http,ftp…etc. So Dionaea can capture the attacks of these types.

```
Activities  Terminal                                                    Fri 14:33
                                                          wissal@ubuntu: /opt/dionaea/etc/dionaea
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                                                    dionaea.cfg

# SPDX-FileCopyrightText: none
# SPDX-License-Identifier: CC0-1.0

[dionaea]
download.dir=var/lib/dionaea/binaries/
modules=curl,python,nfq,emu,pcap
#modules=curl,python,emu
processors=filter_streamdumper,filter_emu

listen.mode=getifaddrs
# listen.addresses=127.0.0.1
 listen.interfaces=ens33

[logging]
default.filename=var/log/dionaea/dionaea.log
default.levels=debug
default.domains=*

errors.filename=var/log/dionaea/dionaea-errors.log
                                                          [ Read 94 lines ]
```

```
errors.filename=var/log/dionaea/dionaea-errors.log
errors.levels=warning,error
errors.domains=*

[processor.filter_emu]
name=filter
config.allow.0.protocols=smbd,epmapper,nfqmirrord,mssqld,http,ftp
next=emu

[processor.filter_streamdumper]
name=filter
config.allow.0.types=accept
config.allow.1.types=connect
config.allow.1.protocols=ftpctrl
config.deny.0.protocols=ftpdata,ftpdatacon,xmppclient
next=streamdumper

[processor.streamdumper]
name=streamdumper
config.path=var/lib/dionaea/bistreams/%Y-%m-%d/

[processor.emu]
name=emu
config.limits.files=3
#512 * 1024
config.limits.filesize=524288
config.limits.sockets=3
config.limits.sustain=120
config.limits.idle=30
config.limits.listen=30
config.limits.cpu=120
#// 1024 * 1024 * 1024
config.limits.steps=1073741824

[module.nfq]
queue=2

[module.nl]
# set to yes in case you are interested in the mac address  of the remote (only works for lan)
lookup_ethernet_addr=no
```

```
[module.python]
imports=dionaea.log,dionaea.services,dionaea.ihandlers
sys_paths=default
service_configs=etc/dionaea/services-enabled/*.yaml
ihandler_configs=etc/dionaea/ihandlers-enabled/*.yaml

[module.pcap]
any.interface=any

[services]
bind="0.0.0.0"
modules=http,ftp,smb,mysql,mongodb,sip,tftp,mssql,mqtt,memcache,upnp
```

1.2. Cowrie Installation and Configuration :

Cowrie : Focuses on emulating SSH and Telnet services, with an emphasis on interaction logging (e.g., commands typed by attackers). Cowrie is typically used for capturing interactive attacks, such as brute-force login attempts, credential harvesting, and command execution within the simulated environment.

Installing necessary python packages :

```
fera@ubuntu:~$ sudo apt install python3 python3-venv python3-pip git
[sudo] password for fera:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
git is already the newest version (1:2.25.1-1ubuntu3.13).
python3-pip is already the newest version (20.0.2-5ubuntu1.11).
The following additional packages will be installed:
  python3.8-venv
The following NEW packages will be installed:
  python3-venv python3.8-venv
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Need to get 6,680 B of archives.
After this operation, 38.9 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python3.8-venv amd64
3.8.10-0ubuntu1~20.04.13 [5,452 B]
Get:2 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 python3-venv amd64 3.8.2-0ubu
ntu2 [1,228 B]
Fetched 6,680 B in 1s (13.2 kB/s)
Selecting previously unselected package python3.8-venv.
(Reading database ... 295164 files and directories currently installed.)
Preparing to unpack .../python3.8-venv_3.8.10-0ubuntu1~20.04.13_amd64.deb ...
Unpacking python3.8-venv (3.8.10-0ubuntu1~20.04.13) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.8.2-0ubuntu2_amd64.deb ...
Unpacking python3-venv (3.8.2-0ubuntu2) ...
Setting up python3.8-venv (3.8.10-0ubuntu1~20.04.13) ...
Setting up python3-venv (3.8.2-0ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
```

```
(cowrie-env) fera@ubuntu:/opt/cowrie$ sudo apt install python3.9 python3.9-venv python3.9-de
v
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython3.9 libpython3.9-dev libpython3.9-minimal libpython3.9-stdlib python3.9-minimal
Suggested packages:
  python3.9-doc binfmt-support
The following NEW packages will be installed:
  libpython3.9 libpython3.9-dev libpython3.9-minimal libpython3.9-stdlib python3.9
  python3.9-dev python3.9-minimal python3.9-venv
0 upgraded, 8 newly installed, 0 to remove and 3 not upgraded.
Need to get 11.3 MB of archives.
After this operation, 47.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython3.9-minimal
amd64 3.9.5-3ubuntu0~20.04.1 [756 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python3.9-minimal amd
64 3.9.5-3ubuntu0~20.04.1 [2,022 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython3.9-stdlib a
md64 3.9.5-3ubuntu0~20.04.1 [1,778 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython3.9 amd64 3.
9.5-3ubuntu0~20.04.1 [1,714 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython3.9-dev amd6
4 3.9.5-3ubuntu0~20.04.1 [4,126 kB]
```

First, we cloned the repository of Cowrie Honeypot :

```
fera@ubuntu:/opt$ sudo git clone https://github.com/cowrie/cowrie.git
Cloning into 'cowrie'...
remote: Enumerating objects: 18583, done.
remote: Counting objects: 100% (2094/2094), done.
remote: Compressing objects: 100% (350/350), done.
remote: Total 18583 (delta 1955), reused 1744 (delta 1743), pack-reused 16489 (from 3)
Receiving objects: 100% (18583/18583), 10.37 MiB | 557.00 KiB/s, done.
Resolving deltas: 100% (13011/13011), done.
```

Giving the privileges to the current user to get the acces to Cowrie, and then, creating a python virtual environment to run Cowrie :

```
fera@ubuntu:/opt$ sudo chown -R $(whoami):$(whoami) cowrie
fera@ubuntu:/opt$ cd /opt/cowrie
fera@ubuntu:/opt/cowrie$ python3 -m venv cowrie-env
fera@ubuntu:/opt/cowrie$ source cowrie-env/bin/activate
(cowrie-env) fera@ubuntu:/opt/cowrie$
(cowrie-env) fera@ubuntu:/opt/cowrie$ ls
bin                docs          Makefile              requirements-output.txt  src
CHANGELOG.rst      etc           MANIFEST.in           requirements-pool.txt    tox.ini
CONTRIBUTING.rst   honeyfs       pyproject.toml        requirements.txt         var
cowrie-env         INSTALL.rst   README.rst            setup.cfg
docker             LICENSE.rst   requirements-dev.txt  setup.py
(cowrie-env) fera@ubuntu:/opt/cowrie$ pip install --upgrade pip

Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
     |                                | 1.8 MB 892 kB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.0.2
    Uninstalling pip-20.0.2:
      Successfully uninstalled pip-20.0.2
Successfully installed pip-24.3.1
```

Installing all dependencies listed in the requirement.txt file to ensure the smoothly working of the honeypot :

```
(cowrie-env) fera@ubuntu:/opt/cowrie$ pip install -r requirements.txt

Collecting attrs==24.3.0 (from -r requirements.txt (line 1))
  Downloading attrs-24.3.0-py3-none-any.whl.metadata (11 kB)
Collecting bcrypt==4.2.1 (from -r requirements.txt (line 2))
  Downloading bcrypt-4.2.1-cp37-abi3-manylinux_2_28_x86_64.whl.metadata (9.8 kB)
Collecting cryptography==44.0.0 (from -r requirements.txt (line 3))
  Downloading cryptography-44.0.0-cp37-abi3-manylinux_2_28_x86_64.whl.metadata (5.7 kB)
Collecting hyperlink==21.0.0 (from -r requirements.txt (line 4))
  Downloading hyperlink-21.0.0-py2.py3-none-any.whl.metadata (1.5 kB)
Collecting idna==3.10 (from -r requirements.txt (line 5))
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting packaging==24.2 (from -r requirements.txt (line 6))
  Downloading packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pyasn1_modules==0.4.1 (from -r requirements.txt (line 7))
  Downloading pyasn1_modules-0.4.1-py3-none-any.whl.metadata (3.5 kB)
Collecting requests==2.32.3 (from -r requirements.txt (line 8))
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting service_identity==24.2.0 (from -r requirements.txt (line 9))
  Downloading service_identity-24.2.0-py3-none-any.whl.metadata (5.1 kB)
Collecting tftpy==0.8.2 (from -r requirements.txt (line 10))
  Downloading tftpy-0.8.2.tar.gz (34 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
```

We copied the default configurations of cowrie and userdb to new directories :

```
(cowrie-env) fera@ubuntu:/opt/cowrie$ cp etc/userdb.example etc/userdb
(cowrie-env) fera@ubuntu:/opt/cowrie$ cd etc
(cowrie-env) fera@ubuntu:/opt/cowrie/etc$ ls
cowrie.cfg.dist  userdb  userdb.example
(cowrie-env) fera@ubuntu:/opt/cowrie/etc$ cp cowrie.cfg.dist cowrie.cfg
```

The file cowrie.cfg contains the main configuration, it defines various settings related to the honeypot behavior including : listening ports and addresses, logginf files and service simulation (SSH,Telnet), and some security ciphers.

```
  GNU nano 4.8                                /opt/cowrie/etc/cowrie.cfg
[honeypot]
# Basic honeypot configuration
hostname = svr04
log_path = var/log/cowrie
download_path = ${honeypot:state_path}/downloads
state_path = var/lib/cowrie
contents_path = honeyfs
txtcmds_path = txtcmds
ttylog = true
ttylog_path = ${honeypot:state_path}/tty
logtype = rotating
timezone = UTC

[ssh]
# Enable SSH logging
enabled = true
listen_port = 2222  # Change this if 2222 is already in use
listen_addr = 0.0.0.0
ciphers = aes128-ctr,aes256-ctr,aes128-gcm,aes256-gcm

[telnet]
# Enable Telnet logging
enabled = true
listen_port = 2323
listen_addr = 0.0.0.0

[output_textlog]
# Log all activity in a text file
logfile = ${honeypot:log_path}/cowrie.log

[output_jsonlog]
# Save logs in JSON format
logfile = /opt/cowrie/var/log/cowrie/cowrie.json
```

We gives the peermission to the cowrie directories :

```
(cowrie-env) fera@ubuntu:~$ sudo chown -R cowrie:cowrie /opt/cowrie/var/run
(cowrie-env) fera@ubuntu:~$ sudo chown -R cowrie:cowrie /opt/cowrie/var/log
(cowrie-env) fera@ubuntu:~$ sudo chmod -R 755 /opt/cowrie/var/run
(cowrie-env) fera@ubuntu:~$ sudo chmod -R 755 /opt/cowrie/var/log
(cowrie-env) fera@ubuntu:~$ sudo nano /etc/systemd/system/cowrie.service
(cowrie-env) fera@ubuntu:~$ sudo mkdir -p /opt/cowrie/var/run
(cowrie-env) fera@ubuntu:~$ sudo mkdir -p /opt/cowrie/var/log
(cowrie-env) fera@ubuntu:~$ sudo chown -R cowrie:cowrie /opt/cowrie/var/run
(cowrie-env) fera@ubuntu:~$ sudo chown -R cowrie:cowrie /opt/cowrie/var/log
```

The creation of the Cowrie service file, it contains the configuration to run Cowrie as a service (User and Group, working directory, the PID file and so on).

```
  GNU nano 4.8                          /etc/systemd/system/cowrie.service
[Unit]
Description=Cowrie Honeypot
After=network.target

[Service]
User=cowrie
Group=cowrie
WorkingDirectory=/opt/cowrie
ExecStart=/opt/cowrie/bin/cowrie start
ExecStop=/opt/cowrie/bin/cowrie stop
Restart=on-failure
RestartSec=10
Environment=PYTHONUNBUFFERED=1
PIDFile=/opt/cowrie/var/run/cowrie.pid

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

After enabling and restarting the Cowrie service, it's actively running.

```
(cowrie-env) fera@ubuntu:~$ sudo systemctl status cowrie
● cowrie.service - Cowrie Honeypot
     Loaded: loaded (/etc/systemd/system/cowrie.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-12-30 10:48:17 PST; 2min 12s ago
   Main PID: 24734 (twistd)
      Tasks: 1 (limit: 9371)
     Memory: 36.8M
     CGroup: /system.slice/cowrie.service
             └─24734 /opt/cowrie/cowrie-env/bin/python3.9 /opt/cowrie/cowrie-env/bin/twistd>

Dec 30 10:48:17 ubuntu systemd[1]: Started Cowrie Honeypot.
Dec 30 10:48:17 ubuntu cowrie[24726]: Using default Python virtual environment "/opt/cowrie>
Dec 30 10:48:17 ubuntu cowrie[24726]: Starting cowrie: [twistd  --umask=0022 --pidfile=var/>
Dec 30 10:48:19 ubuntu cowrie[24726]: /opt/cowrie/cowrie-env/lib/python3.9/site-packages/tw>
Dec 30 10:48:19 ubuntu cowrie[24726]:   b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
Dec 30 10:48:19 ubuntu cowrie[24726]: /opt/cowrie/cowrie-env/lib/python3.9/site-packages/tw>
Dec 30 10:48:19 ubuntu cowrie[24726]:   b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),
```

1.3. Honeyd Installation and Configuration :

Honeyd: is designed to create virtual honeypot networks that simulate a wide variety of operating systems and services to deceive attackers. It can simulate multiple virtual hosts with different IP addresses and services, which can help to emulate entire networks or segments of a network.

We begin by installing the required dependencies :

```
fera@ubuntu:~$ sudo apt install build-essential libpcap-dev libssl-dev libdumbnet-dev bison
flex
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
libpcap-dev is already the newest version (1.9.1-3ubuntu1.20.04.1).
libssl-dev is already the newest version (1.1.1f-1ubuntu2.23).
Suggested packages:
  bison-doc flex-doc

The following NEW packages will be installed:
  bison flex libdumbnet-dev libdumbnet1 libfl-dev libfl2
0 upgraded, 6 newly installed, 0 to remove and 3 not upgraded.
Need to get 1,073 kB of archives.
After this operation, 3,484 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal/main amd64 flex amd64 2.6.4-6.2 [317 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 libdumbnet1 amd64 1.12-9build
1 [25.4 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 libdumbnet-dev amd64 1.12-9bu
ild1 [56.4 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal/main amd64 bison amd64 2:3.5.1+dfsg-1 [657 k
B]
Get:5 http://us.archive.ubuntu.com/ubuntu focal/main amd64 libfl2 amd64 2.6.4-6.2 [11.5 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal/main amd64 libfl-dev amd64 2.6.4-6.2 [6,316
B]
Fetched 1,073 kB in 13s (79.8 kB/s)
Selecting previously unselected package flex.
(Reading database ... 296155 files and directories currently installed.)
Preparing to unpack .../0-flex_2.6.4-6.2_amd64.deb ...
Unpacking flex (2.6.4-6.2) ...
Selecting previously unselected package libdumbnet1:amd64.
```

Then, Cloning the repository from github :

```
fera@ubuntu:~$ git clone https://github.com/DataSoft/Honeyd.git
Cloning into 'Honeyd'...
remote: Enumerating objects: 2641, done.
remote: Total 2641 (delta 0), reused 0 (delta 0), pack-reused 2641 (from 1)
Receiving objects: 100% (2641/2641), 4.46 MiB | 1.56 MiB/s, done.
Resolving deltas: 100% (1348/1348), done.
fera@ubuntu:~/Honeyd$ ls
analyze.c         fdpass.h            ipfrag.c          pfctl_osfp.c        stats.c
analyze.h         filter.c            ipfrag.h          pf.os               stats.h
arp.c             filter.h            keycount.c        pf_osfp.c           strlcat.c
arp.h             generate_assoc.py   keycount.h        pfvar.h             strlcpy.c
atomicio.c        generateDebs        lex.c             plugins.c           strsep.c
autogen.sh        getopt_long.c       lex.l             plugins_config.c    subsystem.c
ChangeLog         gre.c               LICENSE           plugins_config.h    subsystem.h
command.c         gre.h               log.c             plugins.h           subsystems
compat            histogram.c         log.h             pool.c              tagging.c
condition.c       histogram.h         ltmain.sh         pool.h              tagging.h
condition.h       honeyd.8            Makefile.am       pydatahoneyd.c      tcp.c
config.c          honeyd.c            missing           pydatahoneyd.h      tcp.h
config.ethernet   honeydctl.1         mkinstalldirs     pydataprocessing.c  template.h
config.sample     honeydctl.c         network.c         pydataprocessing.h  udp.c
configure         honeyd.h            network.h         pyextend.c          udp.h
configure.in      honeyd_overload.c   nmap.assoc        pyextend.h          ui.c
daemon.c          honeyd_overload.h   nmap-mac-prefixes pypcap              ui.h
debian            honeydstats.c       nmap-os-db        README              untagging.c
debug.h           honeydstats.h       osfp.c            regress             untagging.h
dhcpclient.c      honeydstats_main.c  osfp.h            router.c            util.c
dhcpclient.h      hooks.c             os-test           router.h            util.h
doc               hooks.h             parse.c           rrdtool.c           webserver
dpkt              hsniff.c            parse.h           rrdtool.h           xprobe2.conf
err.c             hsniff.h            parser.h          sample-config       xprobe_assoc.c
ethernet.c        install-sh          parse.y           scripts             xprobe_assoc.h
ethernet.h        interface.c         personality.c     sha1.c
fdpass.c          interface.h         personality.h     stamp-h.in
```

The **autoreconf -i** command initializes or updates the build system configuration by running autoconf, aclocal, and related tools, ensuring that all necessary files (like configure) are generated or updated for building software from source.

```
fera@ubuntu:~/Honeyd$ autoreconf -i
aclocal: warning: autoconf input should be named 'configure.ac', not 'configure.in'
libtoolize: putting auxiliary files in '.'.
libtoolize: copying file './ltmain.sh'
libtoolize: Consider adding 'AC_CONFIG_MACRO_DIRS([m4])' to configure.in,
libtoolize: and rerunning libtoolize and aclocal.
libtoolize: Consider adding '-I m4' to ACLOCAL_AMFLAGS in Makefile.am.
aclocal: warning: autoconf input should be named 'configure.ac', not 'configure.in'
automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
configure.in:8: warning: AM_INIT_AUTOMAKE: two- and three-arguments forms are deprecated.  F
or more info, see:
configure.in:8: https://www.gnu.org/software/automake/manual/automake.html#Modernize-AM_005f
INIT_005fAUTOMAKE-invocation
configure.in:12: installing './compile'
configure.in:5: installing './config.guess'
configure.in:5: installing './config.sub'
Makefile.am:135: warning: 'INCLUDES' is the old name for 'AM_CPPFLAGS' (or '*_CPPFLAGS')
automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
Makefile.am:151: warning: source file 'subsystems/proxy.c' is in a subdirectory,
Makefile.am:151: but option 'subdir-objects' is disabled
automake: warning: possible forward-incompatibility.
automake: At least a source file is in a subdirectory, but the 'subdir-objects'
```

The command **./autogen.sh** is typically used to generate the necessary configuration files for a project, such as configure scripts, by running the project's autoconf and automake tools. This step is often required before compiling and installing the software from source.

```
fera@ubuntu:~/Honeyd$ ./autogen.sh
aclocal: warning: autoconf input should be named 'configure.ac', not 'configure.in'
libtoolize: Consider adding 'AC_CONFIG_MACRO_DIRS([m4])' to configure.in,
libtoolize: and rerunning libtoolize and aclocal.
libtoolize: Consider adding '-I m4' to ACLOCAL_AMFLAGS in Makefile.am.
automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
configure.in:8: warning: AM_INIT_AUTOMAKE: two- and three-arguments forms are deprecated.  F
or more info, see:
configure.in:8: https://www.gnu.org/software/automake/manual/automake.html#Modernize-AM_005f
INIT_005fAUTOMAKE-invocation
Makefile.am:135: warning: 'INCLUDES' is the old name for 'AM_CPPFLAGS' (or '*_CPPFLAGS')
automake: warning: autoconf input should be named 'configure.ac', not 'configure.in'
Makefile.am:151: warning: source file 'subsystems/proxy.c' is in a subdirectory,
Makefile.am:151: but option 'subdir-objects' is disabled
automake: warning: possible forward-incompatibility.
automake: At least a source file is in a subdirectory, but the 'subdir-objects'
automake: automake option hasn't been enabled.  For now, the corresponding output
automake: object file(s) will be placed in the top-level directory.  However,
automake: this behaviour will change in future Automake versions: they will
automake: unconditionally cause object files to be placed in the same subdirectory
automake: of the corresponding sources.
automake: You are advised to start using 'subdir-objects' option throughout your
automake: project, to avoid future incompatibilities.
Makefile.am:151: warning: source file 'subsystems/proxy_main.c' is in a subdirectory,
Makefile.am:151: but option 'subdir-objects' is disabled
Makefile.am:151: warning: source file 'subsystems/smtp.c' is in a subdirectory,
checking for strlcpy... no
checking for strlcat... no
checking for getopt_long... yes
checking for SHA1Update... no
checking for msg_accrights field in struct msghdr... no
checking for sun_len in socketaddr_un... no
checking for msg_control field in struct msghdr... yes
checking for timeradd in sys/time.h... yes
checking for isblank in ctype.h... yes
checking for working addr_cmp in libdnet... yes
checking for addr_net in libdnet... yes
checking for struct sockaddr_storage... yes
checking for sa_len in sockaddr struct... no
checking if underscores are needed for symbols... no
checking if we can access libc without dlopen... no
checking if we can access libc with libc.so... no
checking if we can access libc with libc.so.6... yes
checking for plugins to build in... none
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating regress/Makefile
config.status: creating pypcap/Makefile
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
```

The command ./configure is typically used in the process of compiling software from source code.

```
fera@ubuntu:~/Honeyd$ ./configure
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking target system type... x86_64-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
/home/fera/Honeyd/missing: Unknown `--is-lightweight' option
Try `/home/fera/Honeyd/missing --help' for more information
configure: WARNING: 'missing' script is too old or missing
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking how to print strings... printf
checking whether make supports the include directive... yes (GNU style)
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking dependency style of gcc... none
checking for a sed that does not truncate output... /usr/bin/sed
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for fgrep... /usr/bin/grep -F
checking for ld used by gcc... /usr/bin/ld
```

We used the command : **sudo make install**, for the installation of Honeyd :

```
fera@ubuntu:~/Honeyd$ sudo make install
Making install in .
make[1]: Entering directory '/home/fera/Honeyd'
make[2]: Entering directory '/home/fera/Honeyd'
 /usr/bin/mkdir -p '/usr/bin'
  /bin/bash ./libtool   --mode=install /usr/bin/install -c honeyd honeydctl honeydstats hsni
ff '/usr/bin'
libtool: install: /usr/bin/install -c honeyd /usr/bin/honeyd
libtool: install: /usr/bin/install -c honeydctl /usr/bin/honeydctl
libtool: install: /usr/bin/install -c honeydstats /usr/bin/honeydstats
libtool: install: /usr/bin/install -c hsniff /usr/bin/hsniff
/usr/bin/mkdir -p "/usr/share/honeyd"
(cd . && tar -cf - ./webserver) | \
(cd /usr/share/honeyd && tar -xf -)
find /usr/share/honeyd/webserver -type f | xargs chmod a+r
find /usr/share/honeyd/webserver -type d | xargs chmod a+xr
(cd . && tar -cf - ./scripts) | \
(cd /usr/share/honeyd && tar -xf -)
python /usr/share/honeyd/scripts/lib/init.py
 /usr/bin/mkdir -p '/usr/share/honeyd'
 /usr/bin/install -c -m 644 README nmap.assoc xprobe2.conf nmap-os-db config.sample config.e
thernet pf.os nmap-mac-prefixes '/usr/share/honeyd'
 /usr/bin/mkdir -p '/usr/share/honeyd'
 /usr/bin/mkdir -p '/usr/lib/honeyd'
 /usr/bin/install -c -m 644 libhoneyd.so '/usr/lib/honeyd'
 /usr/bin/mkdir -p '/usr/share/man/man1'
 /usr/bin/install -c -m 644 honeydctl.1 '/usr/share/man/man1'
 /usr/bin/mkdir -p '/usr/share/man/man8'
 /usr/bin/install -c -m 644 honeyd.8 '/usr/share/man/man8'
 /usr/bin/mkdir -p '/usr/include/honeyd'
 /usr/bin/install -c -m 644 hooks.h plugins.h plugins_config.h debug.h '/usr/include/honeyd'
make[2]: Leaving directory '/home/fera/Honeyd'
make[1]: Leaving directory '/home/fera/Honeyd'
```

Honeyd.conf file is the main configuration file used by honeypot software to define the behavior, settings and charactiristics of honeypot (IP addresse, port configuration,….etc).

```
  GNU nano 4.8                                  /opt/honeyd/honeyd.conf
create default
set default default tcp action open
set default default udp action open
set default default icmp action open

# Open port 80 for HTTP traffic
add default tcp port 80 open

# Open port 22 for SSH traffic
add default tcp port 22 open

bind 192.168.62.132 default
```

Testing the functionality of the Honeyd.

```
fera@ubuntu:~/Honeyd$ sudo /usr/bin/honeyd -d -f /opt/honeyd/honeyd.conf
Honeyd V1.6d Copyright (c) 2002-2007 Niels Provos
honeyd[114941]: started with -d -f /opt/honeyd/honeyd.conf
honeyd[114941]: listening promiscuously on ens33: (arp or ip proto 47 or (udp and src port 6
7 and dst port 68) or (ip )) and not ether src 00:0c:29:06:f9:69
honeyd[114941]: Demoting process privileges to uid 65534, gid 65534
```

The creation of the Honeyd service file, it contains the configuration to run Honeyd as a service (User and Group, working directory, logs file, …etc).

```
  GNU nano 4.8                  /etc/systemd/system/honeyd.service
[Unit]
Description=Honeyd Honeypot Daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/honeyd -d -f /opt/honeyd/honeyd.conf
User=root
Group=root
StandardOutput=append:/var/log/honeyd.log
StandardError=append:/var/log/honeyd.log
Restart=always
RestartSec=5s

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

After restarting and enabling the Honeyd service, as shown below it is actively running.

```
fera@ubuntu:~/Honeyd$ sudo nano /etc/systemd/system/honeyd.service
fera@ubuntu:~/Honeyd$ sudo systemctl daemon-reload
fera@ubuntu:~/Honeyd$ sudo systemctl restart honeyd.service
fera@ubuntu:~/Honeyd$ sudo systemctl status honeyd.service
● honeyd.service - Honeyd Honeypot Daemon
     Loaded: loaded (/etc/systemd/system/honeyd.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-12-31 21:39:08 +01; 5s ago
   Main PID: 114131 (honeyd)
      Tasks: 1 (limit: 9371)
     Memory: 10.5M
     CGroup: /system.slice/honeyd.service
             └─114131 /usr/bin/honeyd -d -f /opt/honeyd/honeyd.conf

Dec 31 21:39:08 ubuntu systemd[1]: Started Honeyd Honeypot Daemon.
Dec 31 21:39:08 ubuntu honeyd[114131]: started with -d -f /opt/honeyd/honeyd.conf
Dec 31 21:39:08 ubuntu honeyd[114131]: listening promiscuously on ens33: (arp or ip proto 4
Dec 31 21:39:08 ubuntu honeyd[114131]: Demoting process privileges to uid 65534, gid 65534
```

IV. Simulation of attack scenarios

1. Simulation Methodology:

The simulation methodology for this project focuses on evaluating the effectiveness of various honeypots (Dionaea, Cowrie, and Honeyd) in detecting common network-based attacks. A range of tools, including Nmap, Ping, Curl, SSH, and Telnet, were used to simulate attack scenarios targeting the honeypots. These tools were chosen because they represent common methods attackers use to scan networks, exploit vulnerabilities, and gain unauthorized access to services. The honeypots were configured to expose vulnerable services, such as SSH and Telnet, that are often targeted by attackers. The goal of this methodology was to observe how each honeypot behaves under different types of network reconnaissance and exploitation attempts, and to gather valuable data on the attacks detected and logged by the honeypots.

2. Description of Tested Scenarios:

The tested scenarios focused on simulating common network-based attacks using tools like Nmap, Ping, Curl, SSH, and Telnet. Nmap was used for network scanning and vulnerability probing, attempting to identify open ports and services that could be targeted. Ping was employed to perform simple ICMP-based reconnaissance to check for active hosts and to attempt Denial-of-Service (DoS) conditions. Curl was used to interact with exposed web services, testing for common web application vulnerabilities like command injection or response to malformed requests. SSH and Telnet were tested with brute-force login attempts and other common attacks, simulating attempts to gain unauthorized access through weak or default credentials. These scenarios provided insights into how effectively each honeypot could detect, log, and respond to different attack techniques commonly used by attackers in the wild.

Dionaea :

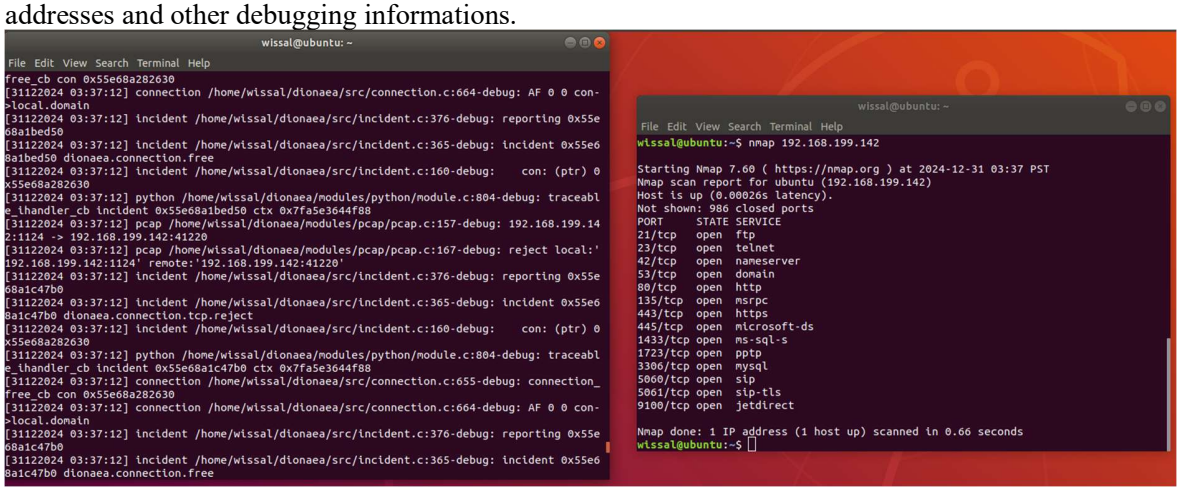
We launched a simple attack for port scanning using nmap :

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.199.137
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-27 18:23 EST
Nmap scan report for 192.168.199.137
Host is up (0.0023s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT    STATE  SERVICE
21/tcp  open   ftp
22/tcp  closed ssh
80/tcp  open   http
443/tcp open   https
445/tcp open   microsoft-ds
MAC Address: 00:0C:29:FC:A9:AF (VMware)

Nmap done: 1 IP address (1 host up) scanned in 5.27 seconds
```

And also we tested the curl :

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ curl http://192.168.199.137:80
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="../">../</a>
</ul>
<hr>
</body>
</html>
```

The image below shows the response of Dionaea to the attacks, including the source and destination IP addresses and other debugging informations.



Cowrie :

We checked that the SSH and telnet ports are listening, so that the attacks of these types will be accepted.

```
fera@ubuntu:~$ sudo netstat -tuln | grep 2222
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN
fera@ubuntu:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
9200                       ALLOW       Anywhere
9200                       ALLOW       198.51.100.0
9200/tcp                   ALLOW       Anywhere
21/tcp                     ALLOW       Anywhere
445/tcp                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
1200/tcp                   ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
9200 (v6)                  ALLOW       Anywhere (v6)
9200/tcp (v6)              ALLOW       Anywhere (v6)
21/tcp (v6)                ALLOW       Anywhere (v6)
445/tcp (v6)               ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
1200/tcp (v6)              ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)

53/udp                     ALLOW OUT   Anywhere
80,443/tcp                 ALLOW OUT   Anywhere
53/udp (v6)                ALLOW OUT   Anywhere (v6)
80,443/tcp (v6)            ALLOW OUT   Anywhere (v6)

fera@ubuntu:~$ sudo ufw allow 2222/tcp
Rule added
Rule added (v6)
```

We launched an SSH attack to view the behavior of the honeypot « Cowrie ».

```
┌──(kali㉿kali)-[~]
└─$ ssh -p 2222 fera@192.168.62.132
The authenticity of host '[192.168.62.132]:2222 ([192.168.62.132]:2222)' can't be establish
ed.
ED25519 key fingerprint is SHA256:Df3XinIOc4CZZPLZb+TBCZlUPx8ETuucHkbI5r8fi00.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.62.132]:2222' (ED25519) to the list of known hosts.
fera@192.168.62.132's password:
Permission denied, please try again.
fera@192.168.62.132's password:


┌──(kali㉿kali)-[~]
└─$ ssh -p 2222 fera@192.168.62.132
fera@192.168.62.132's password:
Permission denied, please try again.
fera@192.168.62.132's password:
Permission denied, please try again.
fera@192.168.62.132's password:
fera@192.168.62.132: Permission denied (publickey,password).
```

We viewed the logs from the json file « cowrie.json ». that shows the different attacks detected by the honeypot cowrie, and informations such as the IP addresses and the passwords entered by the attackers…etc.

V. Integration of security concepts

This section discusses the implementation of security concepts, including Seccomp and AppArmor, and their impact on the honeypot project. These security measures aim to enhance the security of the honeypots and prevent unauthorized access or exploitation of vulnerabilities within the environment.

1. Seccomp Implementation

Seccomp (Secure Computing Mode) is a Linux kernel feature that allows applications to limit the system calls they can make, improving security by reducing the attack surface. In this project, Seccomp was implemented to enhance the security of the honeypot services by restricting their interaction with the kernel. The honeypot services, such as Dionaea, Cowrie, and Honeyd, were configured with Seccomp profiles to only allow a predefined set of system calls necessary for their operation, and block all others that could potentially be exploited by attackers. By limiting system calls, the honeypot's exposure to malicious exploitation was minimized, preventing attackers from executing harmful actions that could compromise the system or gain control of the host. This layer of defense effectively created a more secure

environment for running the honeypots and collecting attack data, without unnecessary permissions being granted to the honeypot processes.

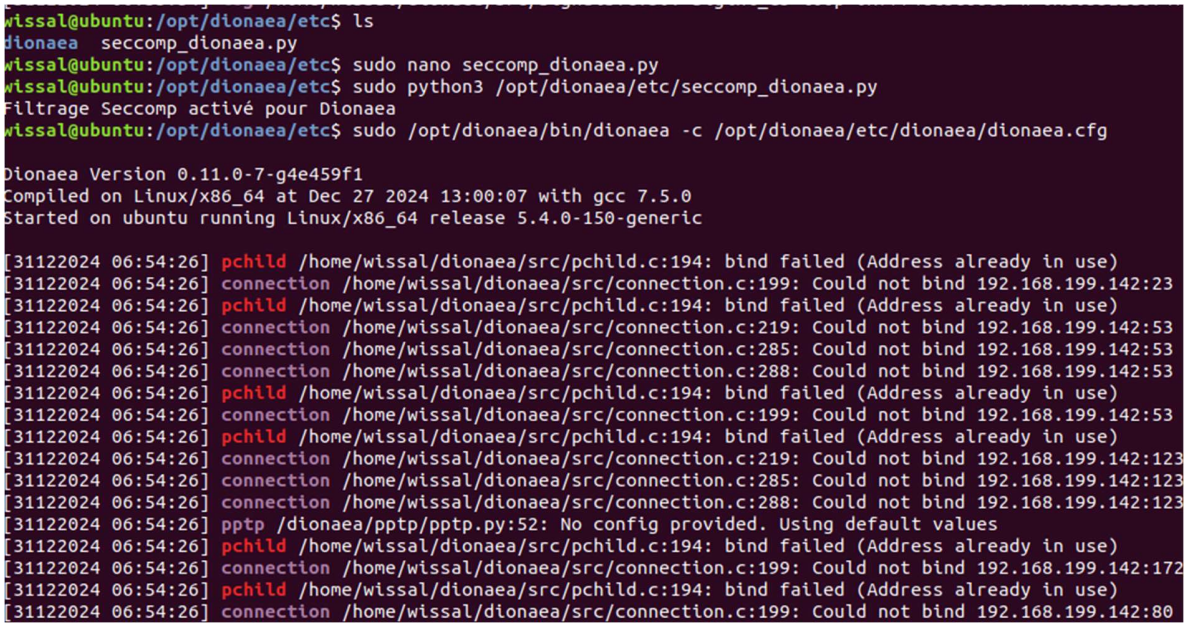
Installation of tools of Seccomp:

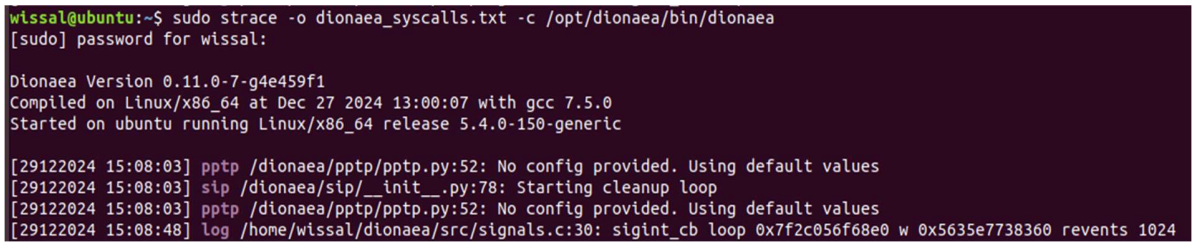
```
wissal@ubuntu:~$ sudo apt install libseccomp-dev gcc make -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
make is already the newest version (4.1-9.1ubuntu1).
make set to manually installed.
gcc is already the newest version (4:7.4.0-1ubuntu2.3).
gcc set to manually installed.
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-goa-1.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0 gir1.2-snap
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1 libcdr-0.1-1 libclucene-contr
  libe-book-0.1-1 libedataserverui-1.2-2 libeot0 libepubgen-0.1-1 libetonyek-0.1-1 libexiv2-14 libfreerdp-client2-2 libfreerdp2-2
  libgpod-common libgpod4 liblangtag-common liblangtag1 liblirc-client0 libmediaart-2.0-0 libmspub-0.1-1 libodfgen-0.1-1 libqqwin
  libsuitesparseconfig5 libvncclient1 libwinpr2-2 libxapian30 libxmlsec1-nss lp-solve media-player-info python3-mako python3-mark
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  seccomp
The following NEW packages will be installed:
  libseccomp-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 81.7 kB of archives.
After this operation, 412 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libseccomp-dev amd64 2.5.1-1ubuntu1~18.04.2 [81.7 kB]
Fetched 81.7 kB in 2s (53.4 kB/s)
Selecting previously unselected package libseccomp-dev:amd64.
(Reading database ... 230871 files and directories currently installed.)
Preparing to unpack .../libseccomp-dev_2.5.1-1ubuntu1~18.04.2_amd64.deb ...
Unpacking libseccomp-dev:amd64 (2.5.1-1ubuntu1~18.04.2) ...
Setting up libseccomp-dev:amd64 (2.5.1-1ubuntu1~18.04.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
wissal@ubuntu:~$
```

We applied a filter to allow the syscalls needed by the Dionaea service to ensure its functionality, using the script below. The syscalls that are not defined in the python script are automatically killed.

```
 GNU nano 2.9.3                                                    seccomp_dionaea.py

import pyseccomp  # Importer pyseccomp avec son alias correct

# Créer un filtre Seccomp avec une action par défaut KILL
f = pyseccomp.SyscallFilter(defaction=pyseccomp.LOG)

# Liste des appels système nécessaires pour Dionaea
allowed_syscalls = [
    "read", "write", "open", "close", "stat", "fstat", "lstat", "poll",
    "lseek", "mmap", "mprotect", "munmap", "brk", "rt_sigaction", "rt_sigprocmask",
    "ioctl", "pread64", "pwrite64", "readv", "writev", "access", "pipe", "pipe2",
    "clone", "fork", "vfork", "execve", "wait4", "exit", "exit_group", "epoll_wait",
    "epoll_ctl", "socket", "connect", "accept", "bind", "listen", "sendto", "recvfrom",
    "setsockopt", "getsockopt", "shutdown", "getpid", "getppid", "getuid", "getgid",
    "gettimeofday", "settimeofday", "clock_gettime", "clock_settime", "select", "recvmsg",
    "sendmsg", "futex", "nanosleep", "getdents", "getdents64","prctl", "getrandom"
]

# Ajouter des règles pour chaque appel système autorisé
for syscall in allowed_syscalls:
    try:
        f.add_rule(pyseccomp.ALLOW, syscall)
    except ValueError as e:
        print(f"Erreur lors de l'ajout de l'appel système '{syscall}': {e}")

# Charger le filtre Seccomp
f.load()

print("Filtrage Seccomp activé pour Dionaea")
```

We executed the python script, and then tested Dionaea by running it in the foreground:





3. Use of AppArmor

AppArmor (Application Armor) is a mandatory access control (MAC) security framework that helps protect applications from being exploited if they are compromised. It works by defining security profiles that specify the allowed actions for programs. Unlike traditional discretionary access control (DAC) systems, AppArmor enforces security policies for applications based on their path, ensuring that even if an attacker gains control over a specific application, the damage they can do is limited by the application's profile.

Enforcing Security

- Enforce mode: In this mode, AppArmor strictly enforces the security policies. If an application tries to perform an action not allowed by its profile, AppArmor will deny the action and log the event.
- Complain mode: In this mode, AppArmor logs the violations without enforcing them. This allows administrators to monitor potential security risks before applying stricter policies.

On system like Ubuntu, we installed AppArmor with the command :

```
wissal@ubuntu:~$ sudo apt install apparmor apparmor-utils -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
apparmor is already the newest version (2.12-4ubuntu5.3).
apparmor set to manually installed.
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-goa-1.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0 gir1.2-snapd-1 gir1.2-udi
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1 libcdr-0.1-1 libclucene-contribs1v5 libclu
  libe-book-0.1-1 libedataserverui-1.2-2 libeot0 libepubgen-0.1-1 libetonyek-0.1-1 libexiv2-14 libfreerdp-client2-2 libfreerdp2-2 libgc1c2 libg
  libgpod-common libgpod4 liblangtag-common liblangtag1 liblirc-client0 libmediaart-2.0-0 libmspub-0.1-1 libodfgen-0.1-1 libqqwing2v5 libraw16
  libsuitesparseconfig5 libvncclient1 libwinpr2-2 libxapian30 libxmlsec1-nss lp-solve media-player-info python3-mako python3-markupsafe syslinu
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  python3-apparmor python3-libapparmor
Suggested packages:
  vim-addon-manager
The following NEW packages will be installed:
  apparmor-utils python3-apparmor python3-libapparmor
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 157 kB of archives.
After this operation, 961 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-libapparmor amd64 2.12-4ubuntu5.3 [26.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python3-apparmor amd64 2.12-4ubuntu5.3 [79.5 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 apparmor-utils amd64 2.12-4ubuntu5.3 [50.7 kB]
Fetched 157 kB in 13s (12.3 kB/s)
Selecting previously unselected package python3-libapparmor.
(Reading database ... 230794 files and directories currently installed.)
Preparing to unpack .../python3-libapparmor_2.12-4ubuntu5.3_amd64.deb ...
Unpacking python3-libapparmor (2.12-4ubuntu5.3) ...
Selecting previously unselected package python3-apparmor.
Preparing to unpack .../python3-apparmor_2.12-4ubuntu5.3_amd64.deb ...
Unpacking python3-apparmor (2.12-4ubuntu5.3) ...
Selecting previously unselected package apparmor-utils.
Preparing to unpack .../apparmor-utils_2.12-4ubuntu5.3_amd64.deb ...
Unpacking apparmor-utils (2.12-4ubuntu5.3) ...
Setting up python3-libapparmor (2.12-4ubuntu5.3) ...
Setting up python3-apparmor (2.12-4ubuntu5.3) ...
Setting up apparmor-utils (2.12-4ubuntu5.3) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

Generation of a profile for Dionaea :

```
fera@ubuntu:~$ sudo aa-genprof /opt/dionaea/bin/dionaea
Writing updated profile for /opt/dionaea/bin/dionaea.
Setting /opt/dionaea/bin/dionaea to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
https://gitlab.com/apparmor/apparmor/wikis/Profiles

Profiling: /opt/dionaea/bin/dionaea

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

Finished generating profile for /opt/dionaea/bin/dionaea.
```

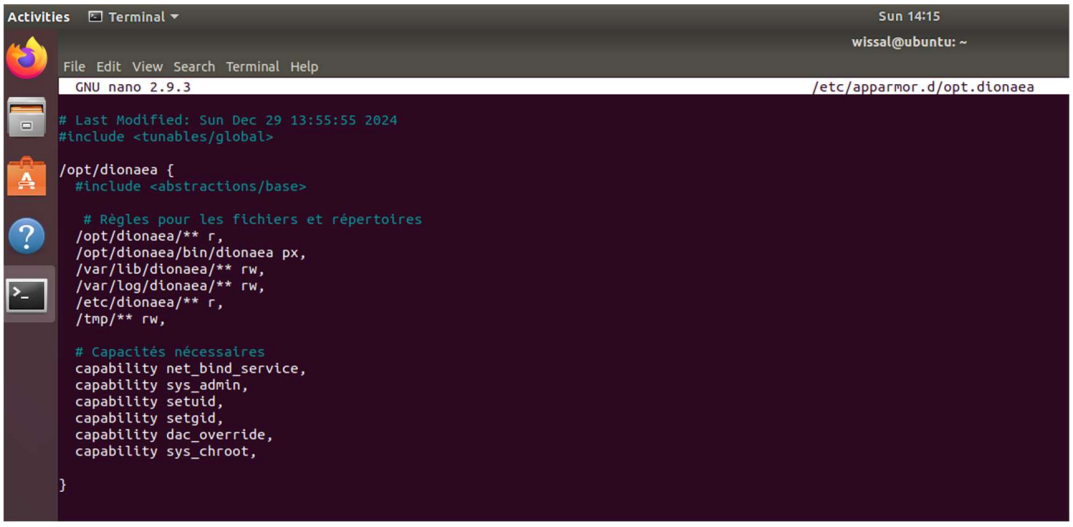
After the generation of the profile, we put the profile in enforced mode in order to deny the actions not permitted and log the events.

```
fera@ubuntu:~$ sudo aa-enforce /etc/apparmor.d/opt.dionaea.bin.dionaea
Setting /etc/apparmor.d/opt.dionaea.bin.dionaea to enforce mode.
fera@ubuntu:~$
```

Viewing the current status of AppArmor profiles :

```
fera@ubuntu:~$ sudo aa-status
apparmor module is loaded.
41 profiles are loaded.
41 profiles are in enforce mode.
   /opt/dionaea/bin/dionaea
   /snap/snapd/23258/usr/lib/snapd/snap-confine
   /snap/snapd/23258/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/bin/evince
   /usr/bin/evince-previewer
   /usr/bin/evince-previewer//sanitized_helper
   /usr/bin/evince-thumbnailer
   /usr/bin/evince//sanitized_helper
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/cups/backend/cups-pdf
   /usr/lib/snapd/snap-confine
   /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/sbin/cups-browsed
   /usr/sbin/cupsd
   /usr/sbin/cupsd//third_party
   /usr/sbin/tcpdump
```

We created a profile for Dionaea on the path of /etc/apparmor.d, this profile defines the rules for the files and repositories and permissions, also defines some capabilities like net_bind_service, sys_admin and so on.



These capabilities are a way of dividing the root user's powerful privileges into smaller, distinct privileges. AppArmor allows you to define which of these capabilities an application can use to reduce the damages.

```
wissal@ubuntu:~$ sudo systemctl status apparmor
● apparmor.service - AppArmor initialization
   Loaded: loaded (/lib/systemd/system/apparmor.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sat 2024-12-28 15:06:32 PST; 12min ago
     Docs: man:apparmor(7)
           http://wiki.apparmor.net/
  Process: 547 ExecStart=/etc/init.d/apparmor start (code=exited, status=0/SUCCESS)
 Main PID: 547 (code=exited, status=0/SUCCESS)

Dec 28 15:06:32 ubuntu systemd[1]: Starting AppArmor initialization...
Dec 28 15:06:32 ubuntu apparmor[547]:  * Starting AppArmor profiles
Dec 28 15:06:32 ubuntu apparmor[547]: Skipping profile in /etc/apparmor.d/disable: usr.bin.firefox
Dec 28 15:06:32 ubuntu apparmor[547]: Skipping profile in /etc/apparmor.d/disable: usr.sbin.rsyslogd
Dec 28 15:06:32 ubuntu apparmor[547]:    ...done.
Dec 28 15:06:32 ubuntu systemd[1]: Started AppArmor initialization.
```

We added two capabilities **cap_net_admin** and **cap_net_raw** ans as a result the dionaea is running without using the sudo.

```
wissal@ubuntu:~$ sudo setcap cap_net_raw,cap_net_admin=eip /opt/dionaea/bin/dionaea
wissal@ubuntu:~$ getcap /opt/dionaea/bin/dionaea
/opt/dionaea/bin/dionaea = cap_net_admin,cap_net_raw+eip
wissal@ubuntu:~$ /opt/dionaea/bin/dionaea -c /opt/dionaea/etc/dionaea/dionaea.cfg

Dionaea Version 0.11.0-7-g4e459f1
Compiled on Linux/x86_64 at Dec 27 2024 13:00:07 with gcc 7.5.0
Started on ubuntu running Linux/x86_64 release 5.4.0-150-generic
```

The parser reads profiles written in plain text (stored in /etc/apparmor.d/) and ensures they are syntactically correct. It interprets the directives, file permissions, and capabilities defined in the profiles. Errors in syntax or unsupported directives will be flagged during parsing.

```
wissal@ubuntu:/etc/apparmor.d$ sudo apparmor_parser -r /etc/apparmor.d/opt.dionaea
wissal@ubuntu:/etc/apparmor.d$ sudo aa-enforce /etc/apparmor.d/opt.dionaea
Setting /etc/apparmor.d/opt.dionaea to enforce mode.
wissal@ubuntu:/etc/apparmor.d$ sudo journalctl -e | grep apparmor
Dec 31 09:18:49 ubuntu sudo[5127]:   wissal : TTY=pts/0 ; PWD=/etc/apparmor.d ; USER=root ; COMMAND=/bin/nano opt.dionaea
Dec 31 09:19:51 ubuntu sudo[5135]:   wissal : TTY=pts/0 ; PWD=/etc/apparmor.d ; USER=root ; COMMAND=/bin/nano opt.dionaea
Dec 31 10:40:02 ubuntu sudo[5323]:   wissal : TTY=pts/0 ; PWD=/etc/apparmor.d ; USER=root ; COMMAND=/sbin/apparmor_parser -r /etc/apparmor.d/opt.dionaea
Dec 31 10:40:02 ubuntu audit[5325]: AVC apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/opt/dionaea" pid=
Dec 31 10:40:02 ubuntu kernel: audit: type=1400 audit(1735670402.993:54): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="un
d=5325 comm="apparmor_parser"
Dec 31 10:40:23 ubuntu sudo[5328]:   wissal : TTY=pts/0 ; PWD=/etc/apparmor.d ; USER=root ; COMMAND=/usr/sbin/aa-enforce /etc/apparmor.d/opt.dionaea
Dec 31 10:40:25 ubuntu audit[5331]: AVC apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/opt/dionaea" pid=
Dec 31 10:40:25 ubuntu kernel: audit: type=1400 audit(1735670425.128:55): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="un
d=5331 comm="apparmor_parser"
Dec 31 10:41:21 ubuntu sudo[5359]:   wissal : TTY=pts/0 ; PWD=/etc/apparmor.d ; USER=root ; COMMAND=/bin/journalctl -e
wissal@ubuntu:/etc/apparmor.d$
```

We ensured that auditd is running to handle logging and auditing of security-related events. It also provides detailed logging of actions including violations of AppArmor policies.

```
wissal@ubuntu:~$ sudo systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable auditd
wissal@ubuntu:~$ sudo systemctl start auditd
wissal@ubuntu:~$ sudo systemctl status auditd
● auditd.service - Security Auditing Service
   Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2024-12-31 12:10:43 PST; 2min 47s ago
     Docs: man:auditd(8)
           https://github.com/linux-audit/audit-documentation
 Main PID: 2973 (auditd)
    Tasks: 2 (limit: 8198)
   CGroup: /system.slice/auditd.service
           └─2973 /sbin/auditd

Dec 31 12:10:43 ubuntu augenrules[2977]: backlog_wait_time 15000
Dec 31 12:10:43 ubuntu augenrules[2977]: enabled 1
Dec 31 12:10:43 ubuntu augenrules[2977]: failure 1
Dec 31 12:10:43 ubuntu augenrules[2977]: pid 2973
Dec 31 12:10:43 ubuntu augenrules[2977]: rate_limit 0
Dec 31 12:10:43 ubuntu augenrules[2977]: backlog_limit 8192
Dec 31 12:10:43 ubuntu augenrules[2977]: lost 0
Dec 31 12:10:43 ubuntu augenrules[2977]: backlog 1
Dec 31 12:10:43 ubuntu augenrules[2977]: backlog_wait_time 0
Dec 31 12:10:43 ubuntu systemd[1]: Started Security Auditing Service.
wissal@ubuntu:~$
```

Integration of security Concepts on Cowrie Honeypot :

AppArmor :

For **AppArmor** we defined the capabilities needded by Cowrie such as net_admin and chown,…etc, and the paths to directories with the permissions (read, write , execute).

```
(cowrie-env) fera@ubuntu:~$ sudo nano /etc/apparmor.d/usr.bin.cowrie
```

```
  GNU nano 4.8                    /etc/apparmor.d/usr.bin.cowrie
# Last Modified: <Date>
#include <tunables/global>

# Define Cowrie binary path
/opt/cowrie/bin/cowrie {
  # Include basic abstractions
  # Include abstractions for base, Python, and other necessary services
  # (modify or add abstractions as needed for Cowrie)
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/openssl>
  #include <abstractions/python>
  #include <abstractions/ubuntu-browsers.d/multimedia>
  #include <abstractions/user-tmp>

  # Define necessary capabilities
  capability chown,
  capability dac_override,
  capability net_admin,
  capability net_bind_service,
  capability net_raw,
  capability setgid,
  capability setuid,

  # Define network permissions
  network bluetooth raw,
  network inet dgram,
  network inet raw,
  network inet stream,
  network inet6 dgram,
  # Deny write access to system files
  deny /etc/group w,
  deny /etc/passwd w,

  # Allow Cowrie directories and logs
  /opt/cowrie/** r,
  /opt/cowrie/bin/** r,
  /opt/cowrie/var/run/** rw,
  /opt/cowrie/var/log/** rw,
  /opt/cowrie/logs/** rw,

  # Allow Python executable and libraries
  /usr/bin/python3 r,
  /usr/lib/python3.*/** r,
  /usr/bin/python3.9 ixr,

  # Allow access to /dev/tty for terminal interaction
  /dev/tty rw,

  # Allow execution and reading of system binaries
  /usr/bin/dirname rix,
  /usr/bin/grep rix,
  /usr/bin/awk rix,
  /usr/bin/sed rix,
```

Then we applied the apparmor on Cowire Honeypot :

```
(cowrie-env) fera@ubuntu:~$ sudo apparmor_parser -r /etc/apparmor.d/usr.bin.cowrie
(cowrie-env) fera@ubuntu:~$ sudo systemctl restart cowrie
(cowrie-env) fera@ubuntu:~$ sudo journalctl -xe | grep apparmor
Dec 30 10:48:14 ubuntu sudo[24715]:     fera : TTY=pts/3 ; PWD=/home/fera ; USER=root ; COMM
AND=/usr/sbin/apparmor_parser -r /etc/apparmor.d/usr.bin.cowrie
Dec 30 10:48:14 ubuntu audit[24717]: AVC apparmor="STATUS" operation="profile_replace" profi
le="unconfined" name="/opt/cowrie/bin/cowrie" pid=24717 comm="apparmor_parser"
Dec 30 10:48:14 ubuntu audit[24717]: SYSCALL arch=c000003e syscall=1 success=yes exit=76693
a0=5 a1=557b635e2880 a2=12b95 a3=0 items=0 ppid=24716 pid=24717 auid=1000 uid=0 gid=0 euid=0
 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts3 ses=4 comm="apparmor_parser" exe="/usr/sbin/a
pparmor_parser" subj=unconfined key=(null)
(cowrie-env) fera@ubuntu:~$ sudo systemctl status cowrie
● cowrie.service - Cowrie Honeypot
     Loaded: loaded (/etc/systemd/system/cowrie.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-12-30 10:48:17 PST; 2min 12s ago
   Main PID: 24734 (twistd)
      Tasks: 1 (limit: 9371)
     Memory: 36.8M
     CGroup: /system.slice/cowrie.service
             └─24734 /opt/cowrie/cowrie-env/bin/python3.9 /opt/cowrie/cowrie-env/bin/twistd>

Dec 30 10:48:17 ubuntu systemd[1]: Started Cowrie Honeypot.
Dec 30 10:48:17 ubuntu cowrie[24726]: Using default Python virtual environment "/opt/cowrie>
Dec 30 10:48:17 ubuntu cowrie[24726]: Starting cowrie: [twistd  --umask=0022 --pidfile=var/>
Dec 30 10:48:19 ubuntu cowrie[24726]: /opt/cowrie/cowrie-env/lib/python3.9/site-packages/tw>
Dec 30 10:48:19 ubuntu cowrie[24726]:   b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
Dec 30 10:48:19 ubuntu cowrie[24726]: /opt/cowrie/cowrie-env/lib/python3.9/site-packages/tw>
Dec 30 10:48:19 ubuntu cowrie[24726]:   b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),
```

Seccomp :

Before applying Seccomp, we identify which system calls Cowrie requires using strace:

```
fera@ubuntu:/opt/cowrie$ strace -f -o cowrie_syscalls.log -c bash /opt/cowrie/b
in/cowrie start
Using default Python virtual environment "/opt/cowrie/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger co
wrie.python.logfile.logger cowrie ]...
/opt/cowrie/cowrie-env/lib/python3.9/site-packages/twisted/conch/ssh/transport.
py:105: CryptographyDeprecationWarning: TripleDES has been moved to cryptograph
y.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptog
raphy.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-cbc": (algorithms.TripleDES, 24, modes.CBC),
/opt/cowrie/cowrie-env/lib/python3.9/site-packages/twisted/conch/ssh/transport.
py:112: CryptographyDeprecationWarning: TripleDES has been moved to cryptograph
y.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptog
raphy.hazmat.primitives.ciphers.algorithms in 48.0.0.
  b"3des-ctr": (algorithms.TripleDES, 24, modes.CTR),
```

After Cowrie is up and running, the cowrie_syscalls.log file will contain system call information. You can analyze it to identify all system calls made by Cowrie.

```
fera@ubuntu:/opt/cowrie$ less cowrie_syscalls.log
```

0.04	0.000085	21	4		pipe
0.04	0.000077	2	37	1	fcntl
0.03	0.000066	2	33		getgid
0.03	0.000065	1	33		getuid
0.03	0.000064	1	33		geteuid
0.03	0.000063	1	33		getegid
0.03	0.000059	4	13		getcwd
0.02	0.000043	8	5		getrandom
0.02	0.000041	41	1	1	faccessat
0.02	0.000033	1	17		dup2
0.02	0.000033	0	34		getpid

```
 0.02    0.000033           4         7           prlimit64
 0.01    0.000030           6         5           set_tid_address
 0.01    0.000026          13         2           sysinfo
 0.01    0.000022           5         4           rt_sigreturn
 0.01    0.000022           4         5           set_robust_list
 0.01    0.000018           4         4           statx
 0.01    0.000017           3         5           lstat
 0.01    0.000016           8         2           sigaltstack
 0.00    0.000006           3         2           uname
 0.00    0.000005           5         1           getpgrp
 0.00    0.000004           4         1           kill
 0.00    0.000004           2         2           chdir
 0.00    0.000004           0         5         4 readlink
 0.00    0.000004           4         1           getppid
 0.00    0.000004           4         1           sched_getaffinity
 0.00    0.000003           1         3           dup
 0.00    0.000000           0         1           socket
 0.00    0.000000           0         1           bind
```

Then we extract unique system calls allowed by Cowrie:

```
fera@ubuntu:/opt/cowrie$ cat cowrie_syscalls.log | awk '{print $NF}' | sort | uniq > allowed_syscalls.txt
fera@ubuntu:/opt/cowrie$ cat allowed_syscalls.txt
---------------
access
arch_prctl
bind
brk
chdir
clone
close
dup
dup2
epoll_create1
epoll_ctl
execve
faccessat
fcntl
fstat
futex
getcwd
getdents64
getegid
geteuid
getgid
getpgrp
getpid
getppid
getrandom
gettid
getuid
ioctl
kill
lseek
lstat
mmap
mprotect
munmap
openat
pipe
pipe2
pread64
```

To integrate the allowed syscalls from Cowrie with `Seccomp` (Secure Computing Mode) for enhanced security, we define a seccomp filter to only allow these specific system calls.

```
  GNU nano 4.8                          seccomp_cowrie.py                          Modified
import pyseccomp
import os
import time

# List of allowed syscalls (Cowrie-specific syscalls)
allowed_syscalls = [
    "access", "arch_prctl", "bind", "brk", "chdir", "clone", "close", "dup", "dup2", "epoll_create1",
    "epoll_ctl", "execve", "faccessat", "fcntl", "fstat", "futex", "getcwd", "getdents64", "getegid",
    "geteuid", "getgid", "getpgrp", "getpid", "getppid", "getrandom", "gettid", "getuid", "ioctl", "kill",
    "lseek", "lstat", "mmap", "mprotect", "munmap", "openat", "pipe", "pipe2", "pread64", "prlimit64", "read",
    "readlink", "rt_sigaction", "rt_sigprocmask", "rt_sigreturn", "sched_getaffinity", "set_robust_list",
    "set_tid_address", "sigaltstack", "socket", "stat", "statx", "syscall", "sysinfo", "uname", "wait4", "write"
]
```

```
# Creating a seccomp filter context (default action: kill process on syscall violation)
seccomp_filter = pyseccomp.SyscallFilter(defaction=pyseccomp.KILL)

# Adding allowed syscalls to the filter
for syscall in allowed_syscalls:
    try:
        seccomp_filter.add_rule(pyseccomp.ALLOW, syscall)
    except ValueError as e:
        print(f"Error adding syscall {syscall}: {e}")

# Applying the filter
try:
    seccomp_filter.load()
    print("Seccomp filter applied. Program will now restrict syscalls.")
except Exception as e:
    print(f"Failed to apply seccomp filter: {e}")
    exit(1)

time.sleep(10)
```

The default action is to kill the process if it tries an unauthorized syscall. If the program tries to make any other syscalls, the process will be terminated by Seccomp.

```
fera@ubuntu:/opt/cowrie$ python3 seccomp_cowrie.py
Seccomp filter applied. Program will now restrict syscalls.
```

To confirm the seccomp filter is active and killing processes on invalid syscalls, we monitor the process's behavior using strace :

```
fera@ubuntu:/opt/cowrie$ strace -e trace=seccomp python3 seccomp_cowrie.py
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=20864, si_uid=1000, si_status=0, si_utime=0, si_stime=1}
 ---
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=20865, si_uid=1000, si_status=0, si_utime=0, si_stime=0}
 ---
seccomp(SECCOMP_SET_MODE_STRICT, 1, NULL) = -1 EINVAL (Invalid argument)
seccomp(SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_TSYNC, NULL) = -1 EFAULT (Bad address)
seccomp(SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_LOG, NULL) = -1 EFAULT (Bad address)
seccomp(SECCOMP_GET_ACTION_AVAIL, 0, [SECCOMP_RET_LOG]) = 0
seccomp(SECCOMP_GET_ACTION_AVAIL, 0, [SECCOMP_RET_KILL_PROCESS]) = 0
seccomp(SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_SPEC_ALLOW, NULL) = -1 EFAULT (Bad address)
seccomp(SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_NEW_LISTENER, NULL) = -1 EFAULT (Bad address)
seccomp(SECCOMP_GET_NOTIF_SIZES, 0, 0x7ffdb833de72) = 0
seccomp(SECCOMP_SET_MODE_FILTER, 0x10 /* SECCOMP_FILTER_FLAG_??? */, NULL) = -1 EFAULT (Bad address)
seccomp(SECCOMP_SET_MODE_FILTER, 0, {len=65, filter=0x232dbdc0}) = 0
Seccomp filter applied. Program will now restrict syscalls.
```

4. Impact of Security Concepts on the Project

The integration of security concepts like Seccomp and AppArmor had a significant impact on the overall security posture of the honeypots used in this project. These security measures not only provided protection for the system running the honeypots but also created an environment where the honeypots could operate in isolation, reducing the potential for compromise. Seccomp reduced the possible attack vectors by restricting the system calls available to the honeypots, while AppArmor's enforcement of fine-grained access controls limited the honeypots' ability to interact with critical system resources.

VI. Visualization and Analysis with ELK Stack

1- Installation and configuration of ELK Stack

Installation of Elasticsearch :

```
wissal@ubuntu:~$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
[sudo] password for wissal:
Sorry, try again.
[sudo] password for wissal:
OK
wissal@ubuntu:~$ sudo sh -c 'echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" > /etc/apt/sources.list.d/elastic-7.x.list'
wissal@ubuntu:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [102 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [102 kB]
Get:4 https://artifacts.elastic.co/packages/7.x/apt stable InRelease [13.7 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [102 kB]
Get:6 https://artifacts.elastic.co/packages/7.x/apt stable/main i386 Packages [96.8 kB]
Get:7 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 Packages [139 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [297 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 DEP-11 Metadata [212 B]
Get:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [303 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,468 B]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [76.8 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 DEP-11 Metadata [212 B]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [61.9 kB]
Get:15 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:16 http://us.archive.ubuntu.com/ubuntu bionic-backports/main amd64 DEP-11 Metadata [8,156 B]
Get:17 http://us.archive.ubuntu.com/ubuntu bionic-backports/restricted amd64 DEP-11 Metadata [210 B]
Get:18 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [10.1 kB]
Get:19 http://us.archive.ubuntu.com/ubuntu bionic-backports/multiverse amd64 DEP-11 Metadata [216 B]
Fetched 1,317 kB in 8s (163 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
wissal@ubuntu:~$ sudo apt install elasticsearch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-goa-1.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0 gir1.2-snapd-1 gir1.2-udisks-2.0 grilo-plugins-0.3-base gstreamer1.0-gtk3
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1 libcdr-0.1-1 libclucene-contribs1v5 libclucene-core1v5 libcmis-0.5-5v5 libcolamd2 libdazzle-1.0-0
  libe-book-0.1-1 libedataserverui-1.2-2 libeot0 libepubgen-0.1-1 libetonyek-0.1-1 libexiv2-14 libfreerdp-client2-2 libfreerdp2-2 libgc1c2 libgee-0.8-2 libgexiv2-2 libgom-1.0-0 libgpgmepp6
  libgpod-common libgpod4 liblangtag-common liblangtag1 liblirc-client0 liblua5.3-0 libmediaart-2.0-0 libmspub-0.1-1 libodfgen-0.1-1 libqqwing2v5 libraw16 librevenge-0.0-0 libsgutils2-2 libssh-4
  libsuitesparseconfig5 libvncclient1 libwinpr2-2 libxaplan30 libxmlsec1-nss lp-solve media-player-info python3-mako python3-markupsafe syslinux syslinux-common syslinux-legacy usb-creator-common
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 325 MB of archives.
After this operation, 542 MB of additional disk space will be used.
```

Testing that Elasticsearch is working

```
wissal@ubuntu:~$ curl -X GET "localhost:9200/"
{
  "name" : "ubuntu",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "meDXQNveTV-O484vikUALQ",
  "version" : {
    "number" : "7.17.26",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "f40328375bfa289242f942fb3d992032ab662e14",
    "build_date" : "2024-11-28T08:05:55.550508263Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.3",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
wissal@ubuntu:~$
```

Installation of Kibana :

```
  "tagline" : "You Know, for Search"
}
wissal@ubuntu:~$ sudo systemctl start kibana
wissal@ubuntu:~$ sudo systemctl status kibana
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2024-12-27 17:19:11 PST; 49s left
     Docs: https://www.elastic.co
 Main PID: 10614 (node)
    Tasks: 11 (limit: 4620)
   CGroup: /system.slice/kibana.service
           └─10614 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bin/../src/cli/dist --logging.dest=/var/log/kibana/kibana.log --pid.file=/run/kibana/kibana.pid --deprecation.skip_deprecated_se

Dec 27 17:19:11 ubuntu systemd[1]: Started Kibana.
Dec 27 17:19:12 ubuntu kibana[10614]: Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to disable see https://www.elastic.co/guide/en/kibana/7.17/prod
```

Installation of Logstash :

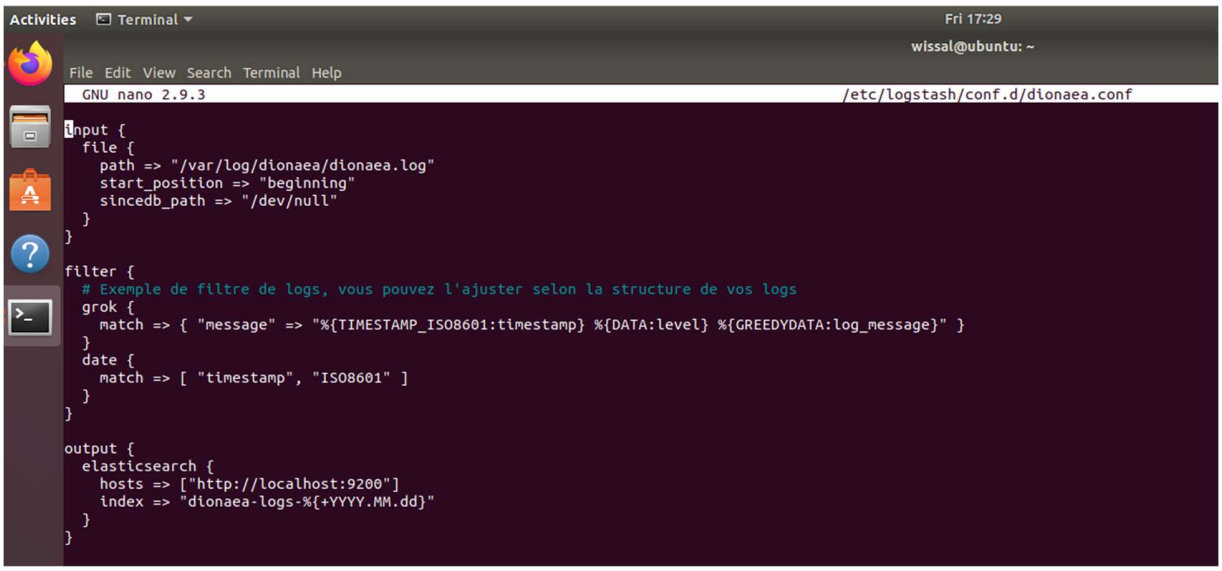
```
wissal@ubuntu:~$ sudo apt install logstash
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-goa-1.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0 gir1.2-snapd-1 gir1.2-udisks-2.0 grilo-plugins-0.3-base gstreamer1.0-gtk3
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1 libcdr-0.1-1 libclucene-contribs1v5 libclucene-core1v5 libcmis-0.5-5v5 libcolamd2 libdazzle-1.0-0
  libe-book-0.1-1 libedataserverui-1.2-2 libeot0 libepubgen-0.1-1 libetonyek-0.1-1 libexiv2-14 libfreerdp-client2-2 libfreerdp2-2 libgc1c2 libgee-0.8-2 libgexiv2-2 libgom-1.0-0 libgpgmepp6
  libgpod-common libgpod4 liblangtag-common liblangtag1 liblirc-client0 liblua5.3-0 libmediaart-2.0-0 libmspub-0.1-1 libodfgen-0.1-1 libqqwing2v5 libraw16 librevenge-0.0-0 libsgutils2-2 libssh-4
  libsuitesparseconfig5 libvncclient1 libwinpr2-2 libxaplan30 libxmlsec1-nss lp-solve media-player-info python3-mako python3-markupsafe syslinux syslinux-common syslinux-legacy usb-creator-common
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 371 MB of archives.
After this operation, 629 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 logstash amd64 1:7.17.26-1 [371 MB]
Fetched 371 MB in 2min 48s (2,204 kB/s)
Selecting previously unselected package logstash.
(Reading database ... 214541 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a7.17.26-1_amd64.deb ...
Unpacking logstash (1:7.17.26-1) ...
Setting up logstash (1:7.17.26-1) ...
Using bundled JDK: /usr/share/logstash/jdk
Using provided startup.options file: /etc/logstash/startup.options
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
```

2- Data collection, indexing and visualization:

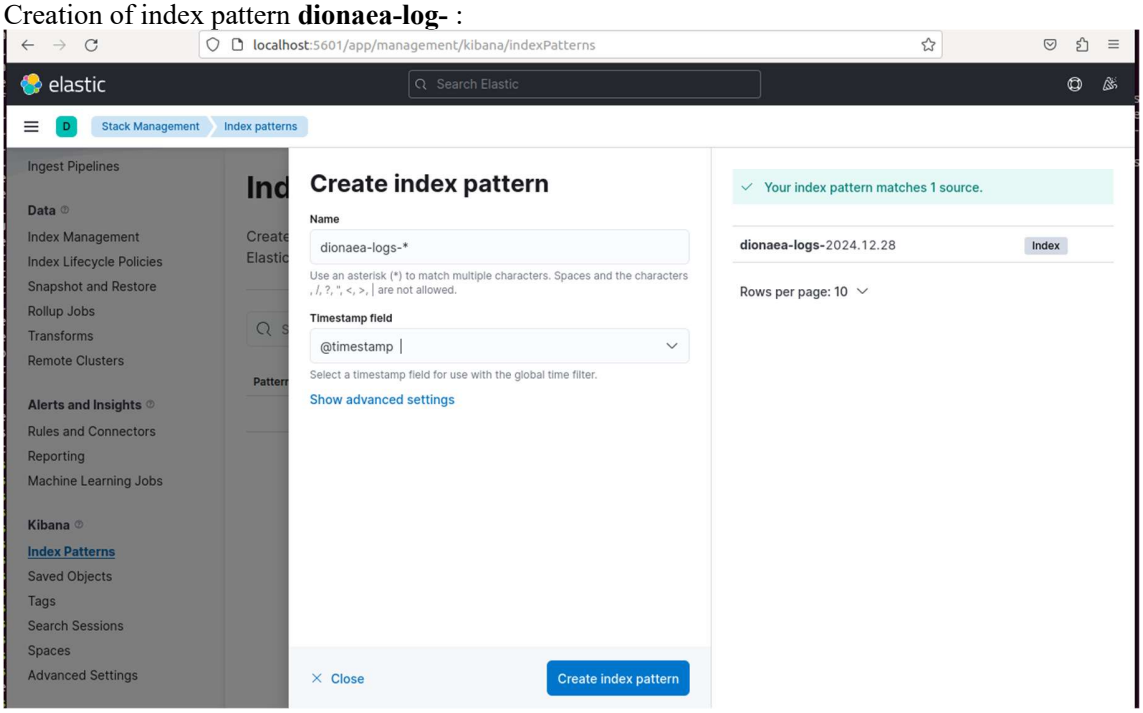
For Dionaea :

The file **/etc/logstash/conf.d/dionaea.conf** plays a key role in configuring Logstash to process and parse logs generated by the Dionaea honeypot. It defines the pipeline that Logstash uses to collect,

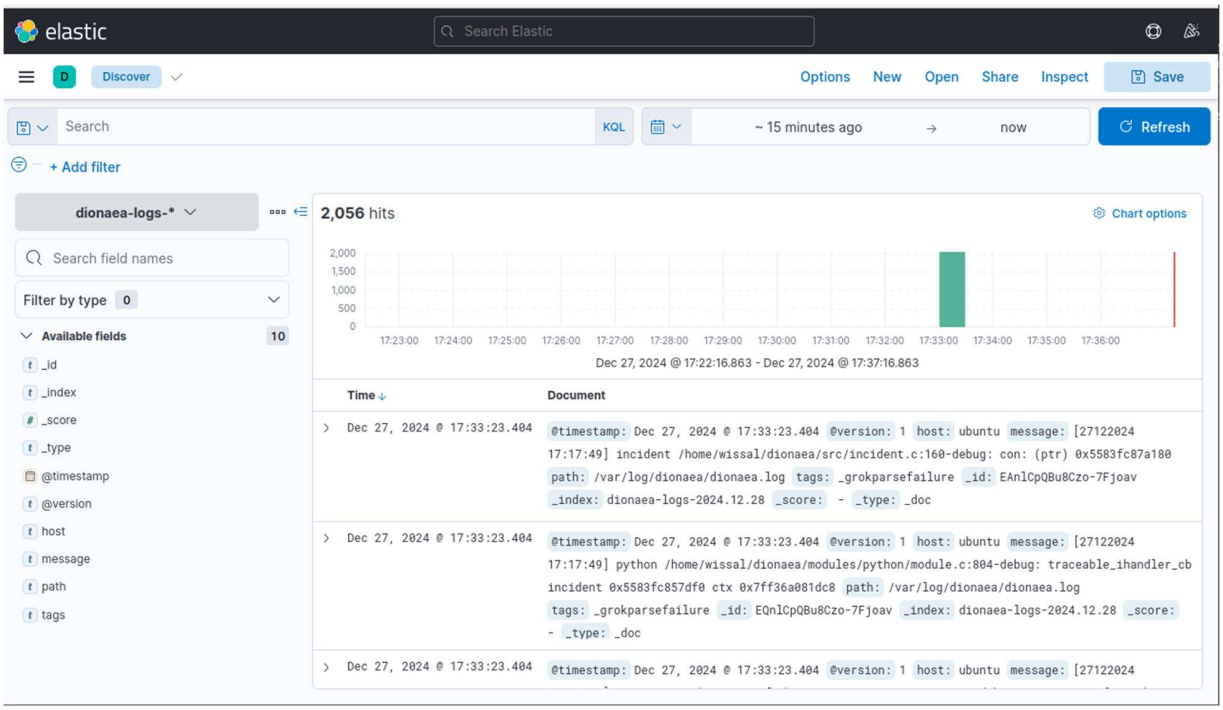
parse, transform, and send Dionaea logs to a destination such as Elasticsearch, a database, or another log storage system.

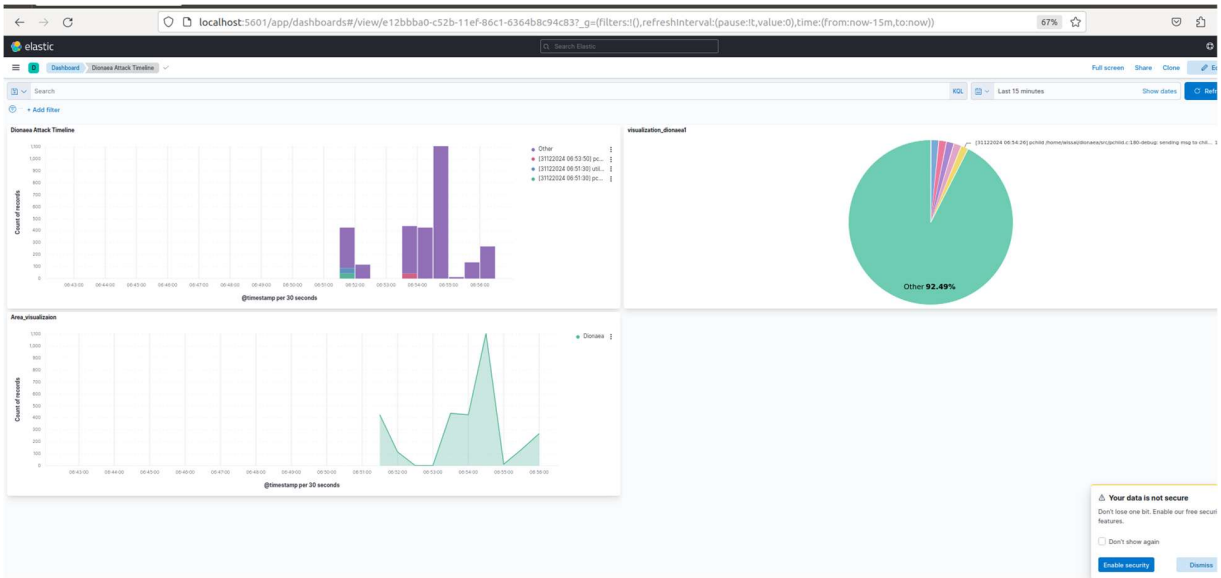


Creation of index pattern **dionaea-log-** :



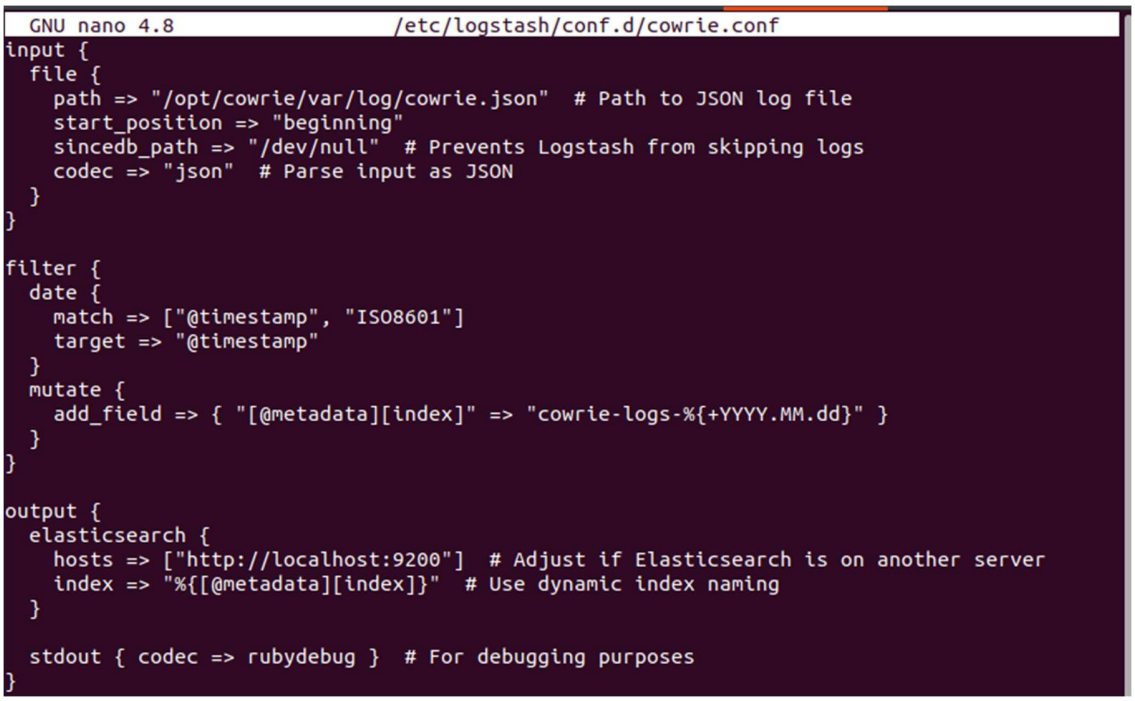
The picture, shows the details about the index pattern, the available fileds and documents with the types of data defined in the /logstash/dionaea.conf.


elastic
Search Elastic
Discover
Options
New
Open
Share
Inspect
Save
Search
KQL
~ 15 minutes ago
now
Refresh
+ Add filter
dionaea-logs-*
2,056 hits
Chart options
Search field names
Filter by type 0
Available fields 10
_id
_index
_score
_type
@timestamp
@version
host
message
path
tags
Dec 27, 2024 @ 17:22:16.863 - Dec 27, 2024 @ 17:37:16.863
Time
Document
Dec 27, 2024 @ 17:33:23.404
@timestamp: Dec 27, 2024 @ 17:33:23.404 @version: 1 host: ubuntu message: [27122024 17:17:49] incident /home/wissal/dionaea/src/incident.c:160-debug: con: (ptr) 0x5583fc87a180 path: /var/log/dionaea/dionaea.log tags: _grokparsefailure _id: EAnlCpQBu8Czo-7Fjoav _index: dionaea-logs-2024.12.28 _score: - _type: _doc
Dec 27, 2024 @ 17:33:23.404
@timestamp: Dec 27, 2024 @ 17:33:23.404 @version: 1 host: ubuntu message: [27122024 17:17:49] python /home/wissal/dionaea/modules/python/module.c:804-debug: traceable_ihandler_cb incident 0x5583fc857df0 ctx 0x7ff36a081dc8 path: /var/log/dionaea/dionaea.log tags: _grokparsefailure _id: EQnlCpQBu8Czo-7Fjoav _index: dionaea-logs-2024.12.28 _score: - _type: _doc
Dec 27, 2024 @ 17:33:23.404
@timestamp: Dec 27, 2024 @ 17:33:23.404 @version: 1 host: ubuntu message: [27122024


localhost:5601/app/dashboards#/view/e12bbba0-c52b-11ef-86c1-6364b8c94c83?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-15m,to:now))
elastic
Dionaea Attack Timeline
visualization_dionaea1
Other 92.49%
Area_visualizaion
Dionaea
Your data is not secure

For Cowrie :

The file **/etc/logstash/conf.d/cowrie.conf** is used to configure Logstash for processing logs generated by the Cowrie honeypot. Cowrie is an SSH and Telnet honeypot that emulates a shell environment to capture and log interactions from attackers. This configuration file defines the pipeline for ingesting, parsing, enriching, and outputting these logs.



```
  GNU nano 4.8                  /etc/logstash/conf.d/cowrie.conf
input {
  file {
    path => "/opt/cowrie/var/log/cowrie.json"  # Path to JSON log file
    start_position => "beginning"
    sincedb_path => "/dev/null"  # Prevents Logstash from skipping logs
    codec => "json"  # Parse input as JSON
  }
}

filter {
  date {
    match => ["@timestamp", "ISO8601"]
    target => "@timestamp"
  }
  mutate {
    add_field => { "[@metadata][index]" => "cowrie-logs-%{+YYYY.MM.dd}" }
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]  # Adjust if Elasticsearch is on another server
    index => "%{[@metadata][index]}"  # Use dynamic index naming
  }

  stdout { codec => rubydebug }  # For debugging purposes
}
```

Input Section

- The source of the logs (e.g., file, syslog, network).

Example: File input to read Cowrie JSON logs from a specific path.

Filter Section

- Processes and structures raw Cowrie logs for analysis.
- Includes JSON parsing, field extraction, and enrichment.

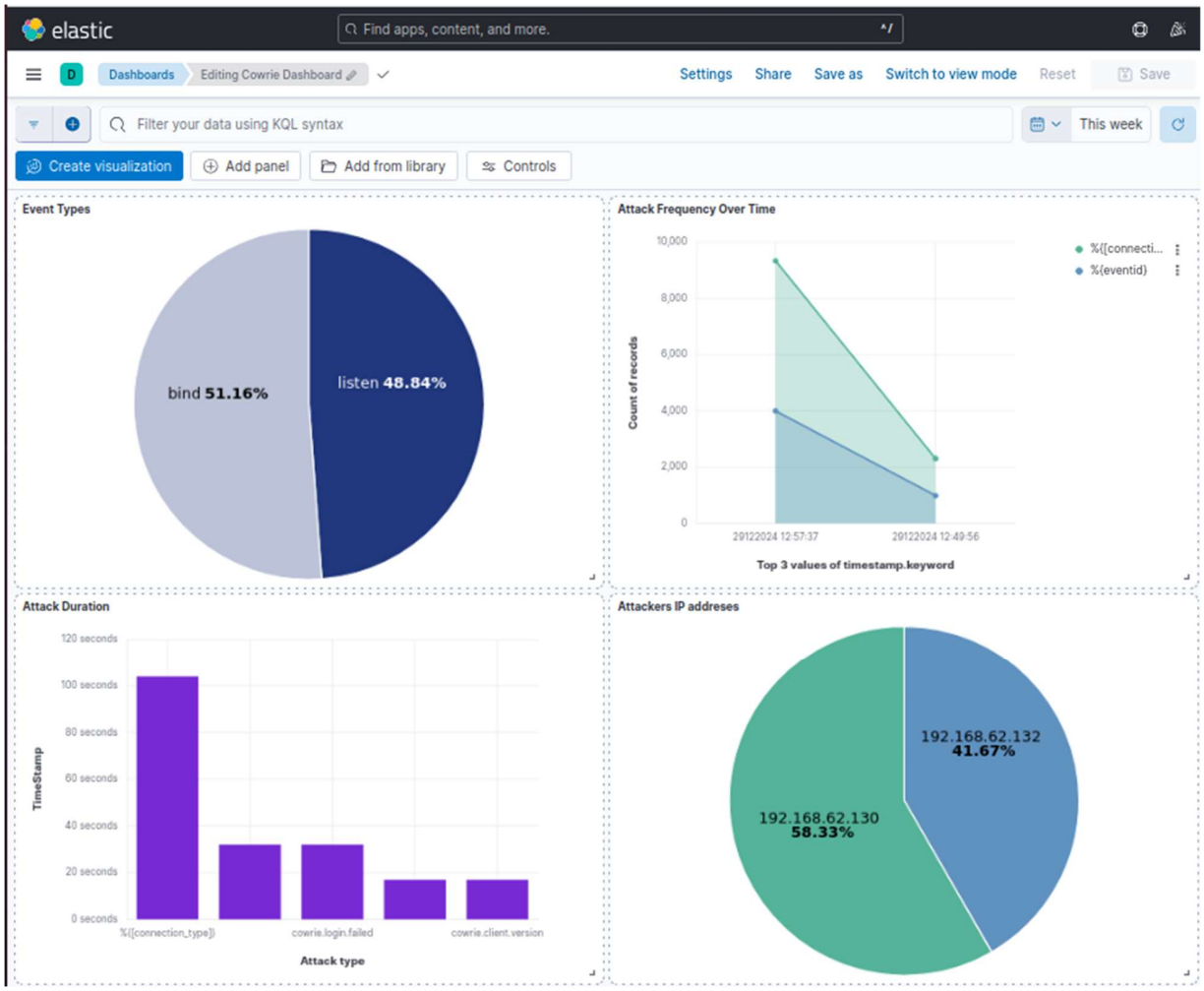
Example: Adding data to determine the attacker's IP.

Output Section

- Defines the destination of processed logs.
- Common outputs: Elasticsearch (for visualization), files (archiving), or alerting systems.

Example: Sending logs to an Elasticsearch index named cowrie-logs-*.

The visualizations show various types of data in simple graphs for ease to understand. There are visualization of event types, Attack frequency over time, Attack duration and the IP addresses of attackers…etc.



VII. Results and discussion

This section evaluates the outcomes of deploying the honeypots, analyzes the data collected during the project, and identifies limitations while proposing potential improvements, with a focus on the security measures implemented using AppArmor and Seccomp.

7.1 Effectiveness of Honeypots

The deployment of Dionaea, Cowrie, and Honeyd demonstrated their value in capturing and analyzing attacker activity while being securely isolated using AppArmor and Seccomp.

- **Dionaea**
  Successfully identified exploitation attempts on common network services like SMB and FTP.
  Security Reinforcement: AppArmor profiles restricted Dionaea's access to critical system resources, while Seccomp limited its system call usage to minimize risks in case of a compromise.
- **Cowrie**
  Recorded multiple brute-force login attempts on the SSH and Telnet protocols.
  Logged attacker interactions, including executed commands and file downloads, providing behavioral insights.
  Security Reinforcement: AppArmor policies ensured that Cowrie could only access designated directories, and Seccomp mitigated risks by blocking unnecessary system calls.
- **Honeyd**
  Simulated diverse operating systems and services, attracting reconnaissance activities and diverting attackers from real assets.
  Generated fake network traffic to enhance realism.

Overall, The honeypots effectively attracted and captured malicious activities while remaining securely isolated from the real environment.

7.2 Analysis of Collected Data

The data collected by the honeypots provided rich insights into attacker methodologies:

Types of Attacks Identified

- **Reconnaissance**: Numerous scans were detected, including port scanning and service enumeration attempts.
- **Credential Attacks**: Over 500 SSH/Telnet brute-force attempts were logged, with commonly used credentials like "admin/admin" or "root/root."
- **Malware Delivery**: Payloads captured by Dionaea included ransomware samples and botnet components targeting SMB and HTTP protocols.

Attacker Behavior

- **Cowrie Logs:** Analysis of command execution patterns revealed attempts to escalate privileges, download additional tools, and establish persistence.
- **Network Activity:** Honeyd logs showed repeated attempts to exploit specific services, reflecting common vulnerabilities attackers probe for in real-world environments.

Visualization with ELK Stack

Using the ELK stack (Elasticsearch, Logstash, and Kibana), the data was aggregated and visualized into dashboards. These included:

- **Attack Frequency:** Graphs showing the frequency of attacks by protocol and service.
- **Credential Analysis:** Tables and pie charts summarizing commonly used usernames and passwords.

7.3 Limitations

a. Limited Attack Scenarios:
The honeypots and security measures were tested with a limited range of simulated attacks (e.g., Nmap, ping, curl, SSH, and Telnet). While effective, these tests do not comprehensively cover all potential real-world attack vectors or sophisticated threats.

b. Honeypot Detectability:
Advanced attackers might detect the presence of honeypots due to their predictable behavior or lack of full emulation of real systems, potentially reducing the effectiveness of capturing meaningful attack data.

7.4 Improvements :

- **Expand Attack Scenarios**:
  Test the honeypots with a wider range of attacks, like malware or ransomware, to see how they perform against different threats.

- **Instant Alerts**:
  Set up alerts to notify administrators immediately when an attack happens.

- **Collect More Data**:
  Run the honeypots longer to gather more real-world attack information for better analysis.

VIII. Challenges faced during the project

- Configuring AppArmor and Seccomp with honeypots like Dionaea was challenging due to:
- Limited documentation on integrating these security frameworks with honeypots.
- AppArmor profiles often required manual adjustments to allow Dionaea to function correctly without overly permissive settings.
- Seccomp's syscalls filtering needed precise configuration to avoid blocking legitimate honeypot operations while maintaining security.

IX. References

- https://medium.com/@noah_h/kubernetes-security-tools-seccomp-apparmor-586fdc61e6d9
- ELK Stack Documentation (Elastic Stack). Available at: https://www.elastic.co/guide/en/elastic-stack/current/index.html
- https://orcacore.com/secure-ubuntu-2404-with-apparmor/
- The official documentation of Honeypots; Dionaea and Cowrie.

X. Conclusion

This project demonstrates the effectiveness of honeypots as a proactive defense mechanism for detecting and analyzing cyberattacks. By deploying tools like Dionaea, Cowrie, and Honeyd, we created a controlled environment to simulate and monitor malicious activities. These honeypots provided valuable insights into attacker behaviors, allowing us to understand and assess vulnerabilities within a network.

The integration of logging and monitoring tools, such as ELK, ensured that captured data could be effectively analyzed, supporting actionable intelligence. While the project successfully highlighted the potential of honeypots in enhancing network security, it also underscored areas for improvement, including expanding attack scenarios, increasing automation, and enhancing the realism of honeypots.

Through this work, we have shown the importance of blending traditional cybersecurity methods with modern techniques to strengthen defenses. This project serves as a foundation for future exploration, emphasizing the need for continuous adaptation to an ever-evolving threat landscape.