2024 - 2025

HONEYPOTS IN ACTION: ANALYZING ATTACKS AND STRENGTHENING DEFENSES

Presented by:
Wissal Boutayeb
Fatima Bouyarmane

Supervised by:
Mr. Airaj Mohammed

PLAN

CONTEXT

As cyber threats grow more advanced, organizations need innovative ways to protect their systems and understand attackers. This project focuses on setting up and securing honeypots—Dionaea, Cowrie, and HoneyD—which act as fake systems to attract and study malicious activities. To make these honeypots more secure, we used AppArmor for access control and Seccomp to filter system calls. This approach not only collects useful threat data but also keeps the honeypots safe from attacks.

PROJECT OBJECTIVES

Simulate Attacks: Use honeypots to attract attackers and observe their behavior, techniques, and methods.

Enhance Security: Apply tools like AppArmor and Seccomp to keep honeypots safe and secure

Data Integration and Visualization: Use the ELK stack (Elasticsearch, Logstash, Kibana) to analyze attack data and create easy-to-understand visuals to improve defenses.

METHODOLOGY AND TOOLS

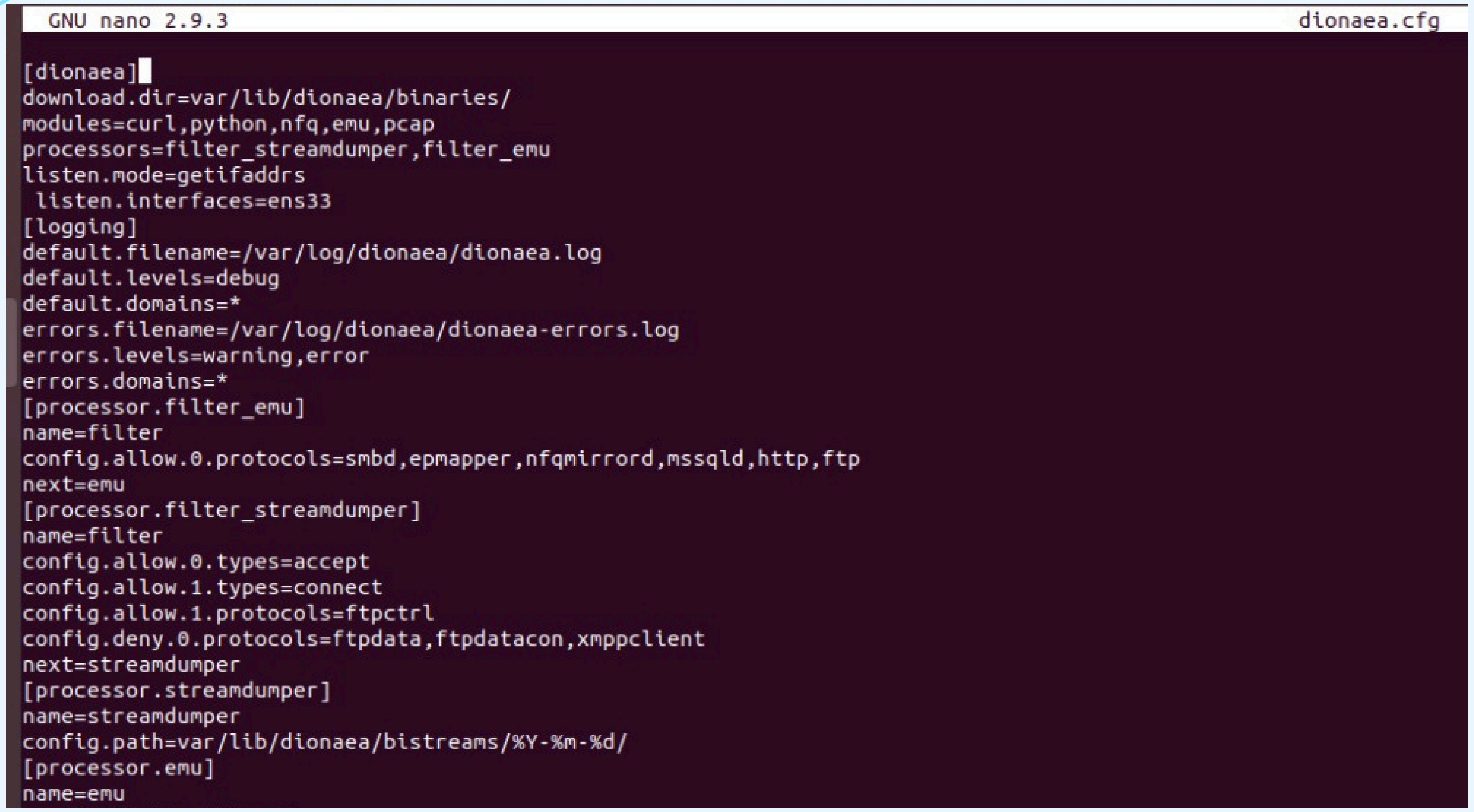
HONEYPOTS OVERVIEW

Dionaea: A low-interaction honeypot designed to capture malware and study its propagation methods. It was deployed to attract and collect malicious payloads from various attack vectors.

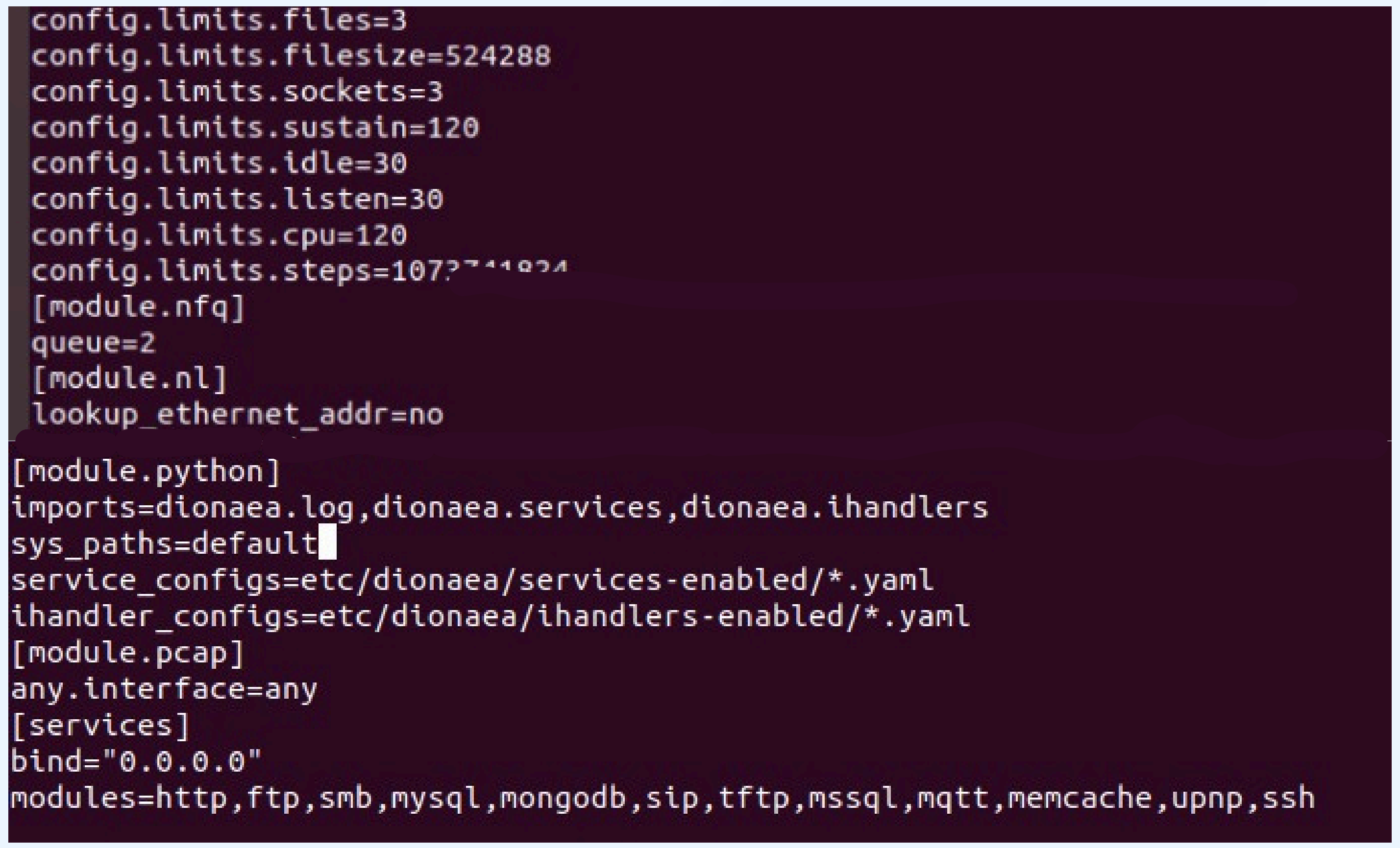
Cowrie: A medium-interaction honeypot that emulates a shell environment, used to monitor and log attackers' commands, behaviors, and attempted exploitations.

Honeyd: A flexible honeypot that can pretend to be a whole network, helping us learn how attackers scan and gather information.

DIONAEA CONFIGURATION

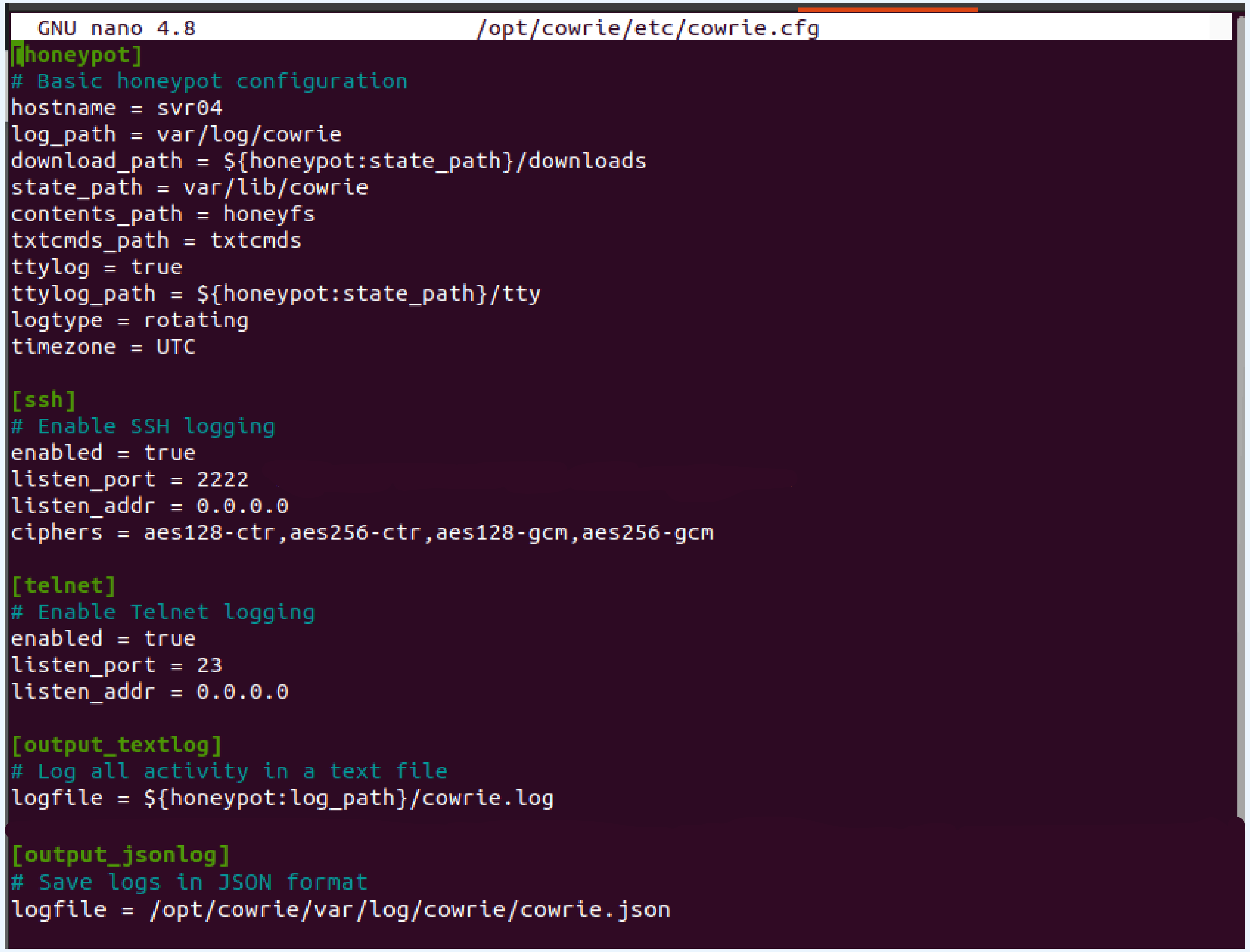


```
  GNU nano 2.9.3                                                                dionaea.cfg

[dionaea]
download.dir=var/lib/dionaea/binaries/
modules=curl,python,nfq,emu,pcap
processors=filter_streamdumper,filter_emu
listen.mode=getifaddrs
 listen.interfaces=ens33
[logging]
default.filename=/var/log/dionaea/dionaea.log
default.levels=debug
default.domains=*
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
```



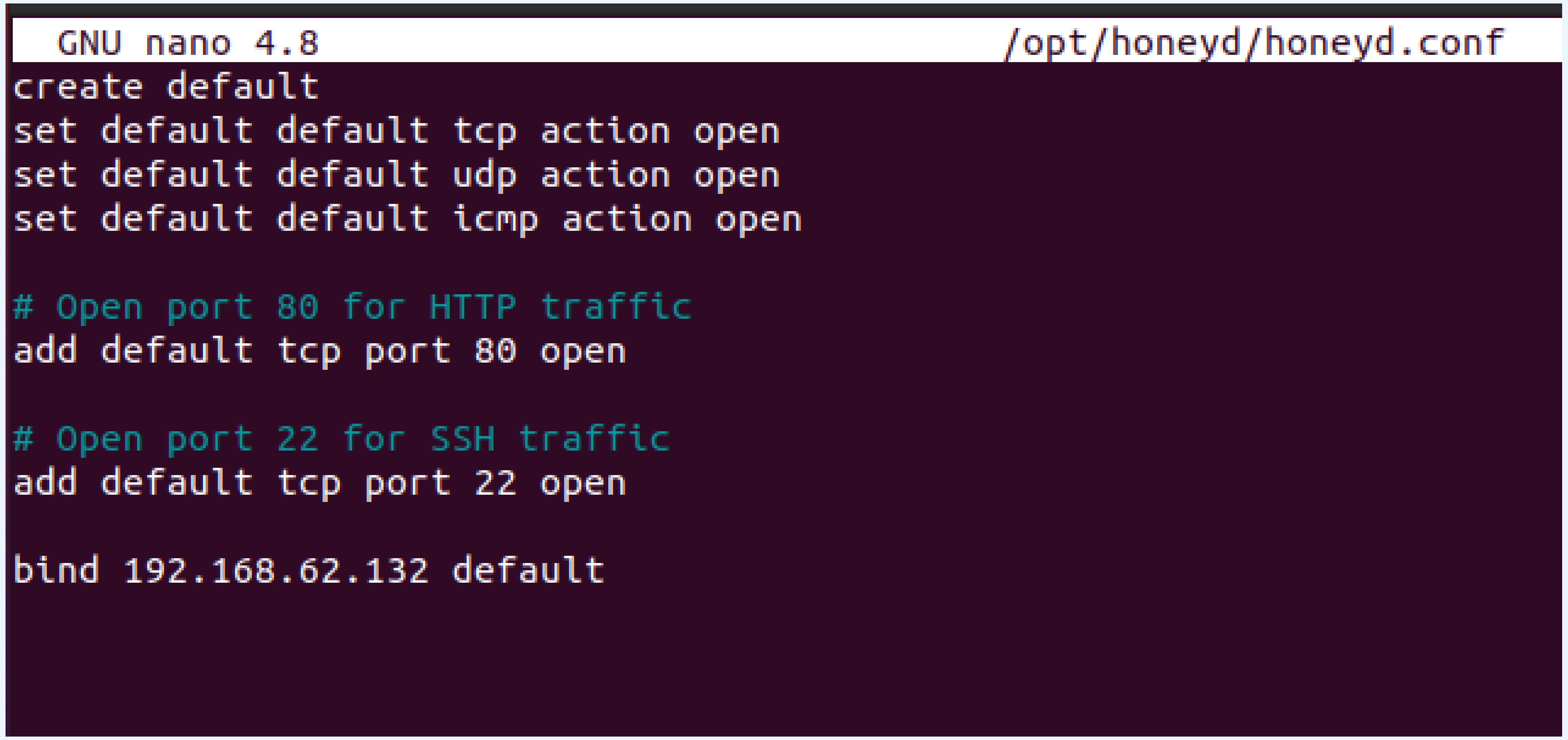
```
 config.limits.files=3
 config.limits.filesize=524288
 config.limits.sockets=3
 config.limits.sustain=120
 config.limits.idle=30
 config.limits.listen=30
 config.limits.cpu=120
 config.limits.steps=107
 [module.nfq]
 queue=2
 [module.nl]
 lookup_ethernet_addr=no
[module.python]
imports=dionaea.log,dionaea.services,dionaea.ihandlers
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,ssh
```

COWRIE CONFIGURATION



```
  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
listen_addr = 0.0.0.0
ciphers = aes128-ctr,aes256-ctr,aes128-gcm,aes256-gcm

[telnet]
# Enable Telnet logging
enabled = true
listen_port = 23
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
```

HONEYD CONFIGURATION



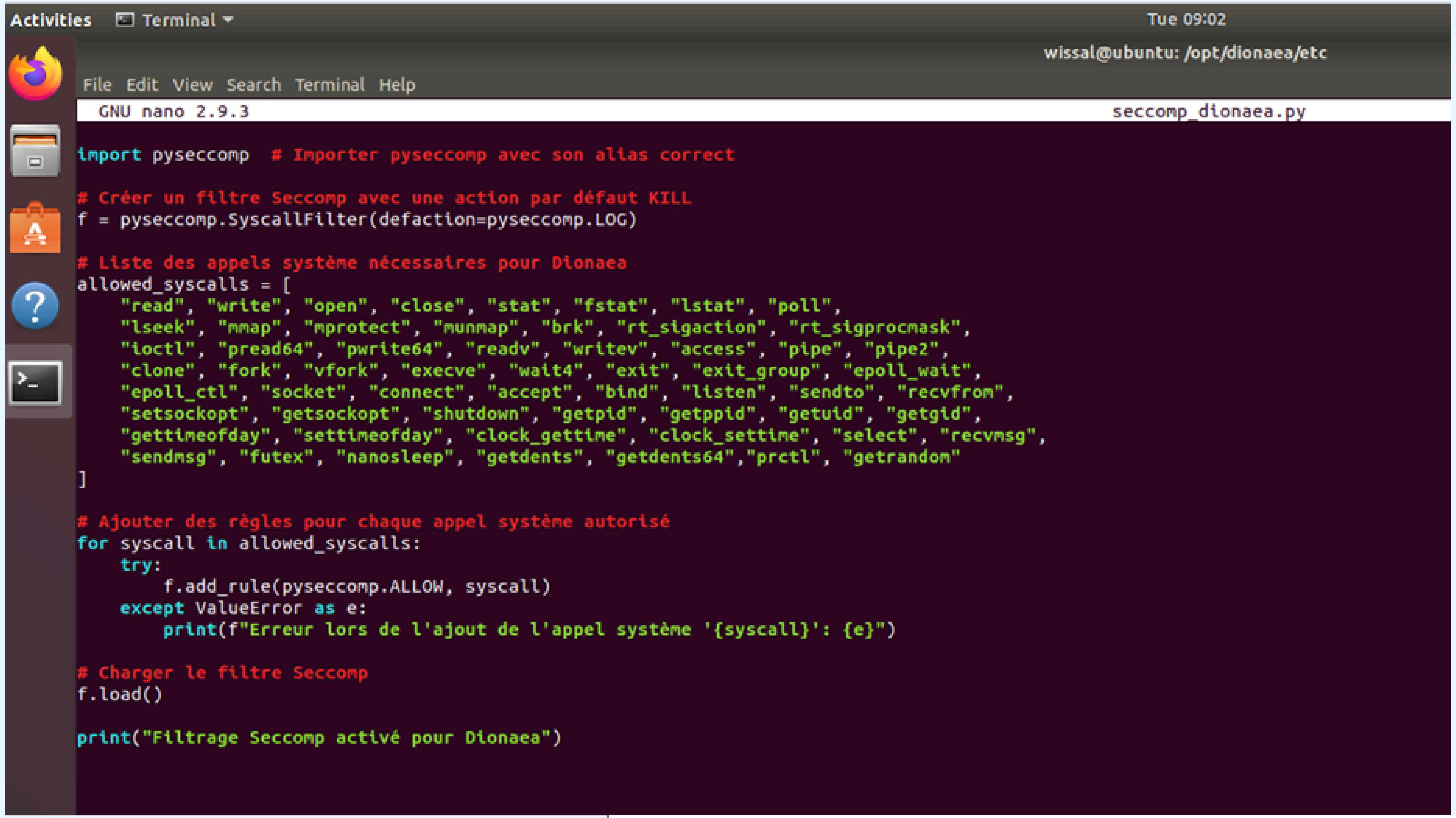
```
  GNU nano 4.8                                    /opt/honeyd/honeyd.conf
create default
set default default tcp action open
set default default udp action open
set default default icmp action open

# 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
```

SECURITY CONCEPTS

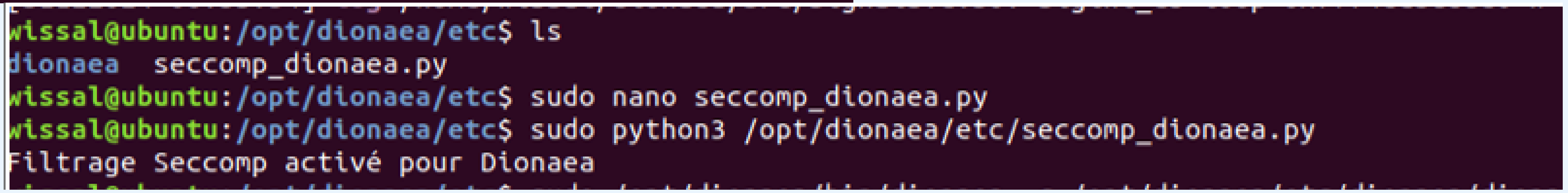
AppArmor: This Linux security module was configured to enforce mandatory access control, restricting the actions of honeypots to predefined policies and minimizing their attack surface.

Seccomp: Used to limit the system calls accessible to honeypot processes, ensuring that even if compromised, the attacker's actions would be severely constrained.

SECURITY CONCEPTS

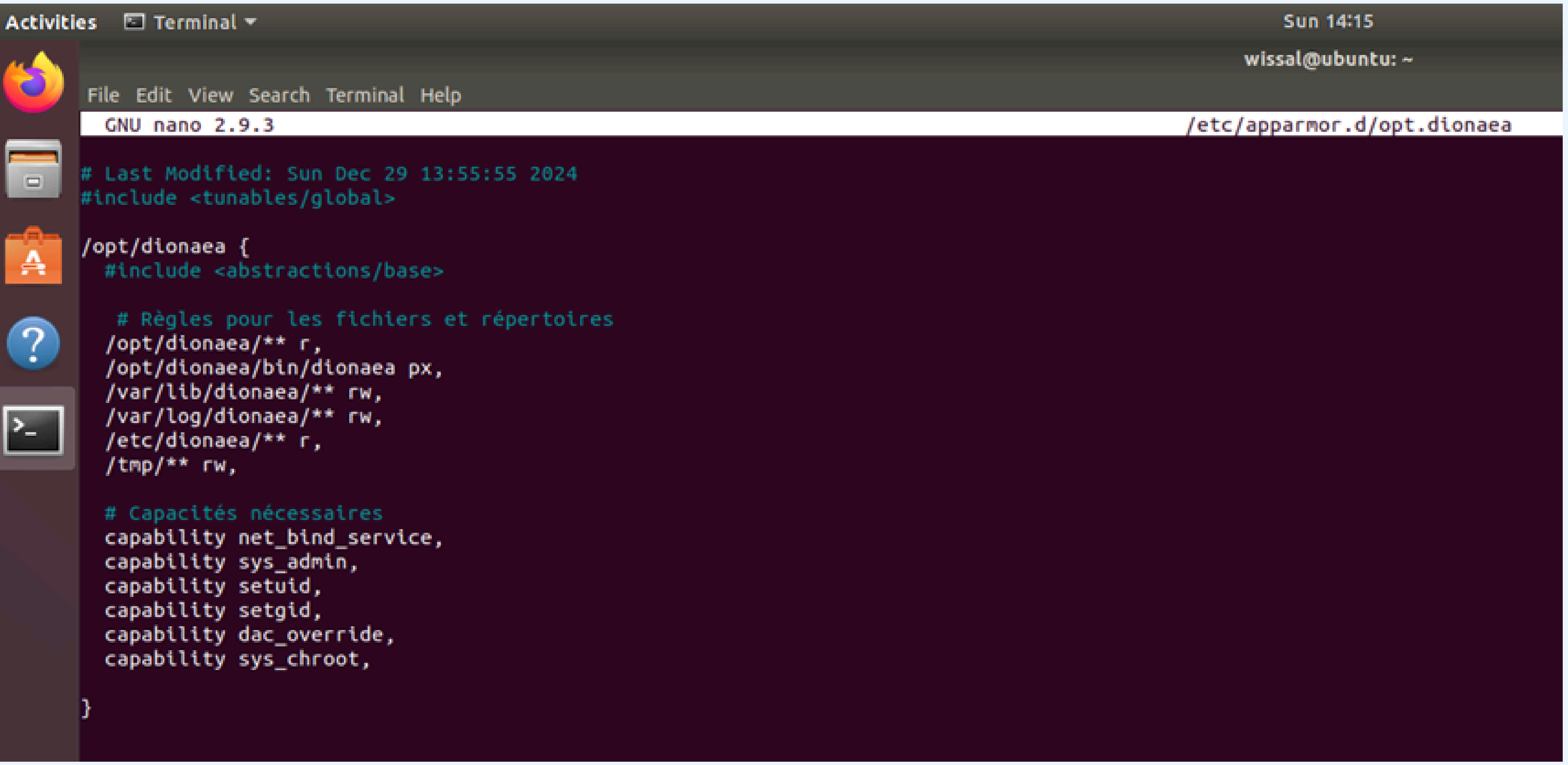

Activities
Terminal
Tue 09:02
wissal@ubuntu: /opt/dionaea/etc
File Edit View Search Terminal Help

```
  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")
```

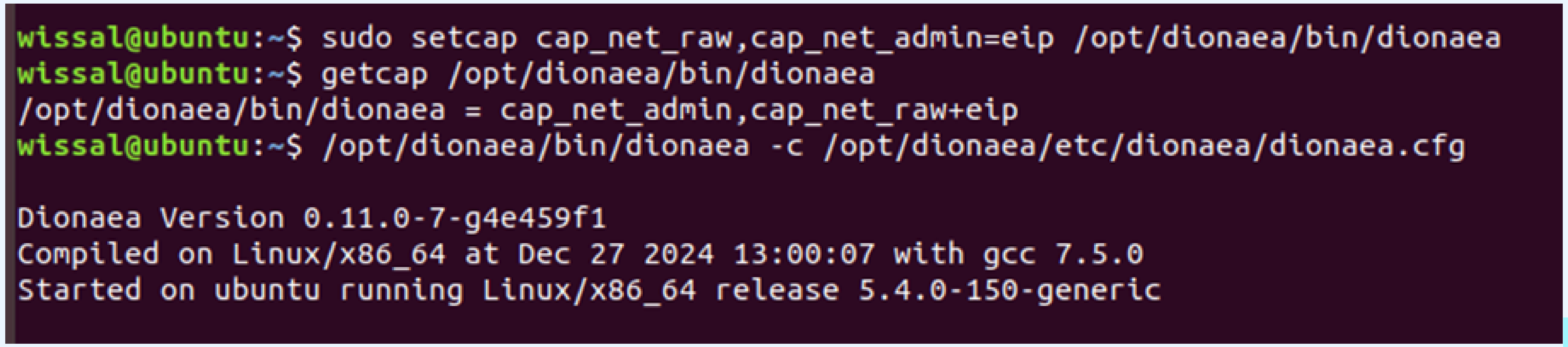


```
wissal@ubuntu:/opt/dionaea/etc$ ls
dionaea  seccomp_dionaea.py
wissal@ubuntu:/opt/dionaea/etc$ sudo nano seccomp_dionaea.py
wissal@ubuntu:/opt/dionaea/etc$ sudo python3 /opt/dionaea/etc/seccomp_dionaea.py
Filtrage Seccomp activé pour Dionaea
```

SECURITY CONCEPTS

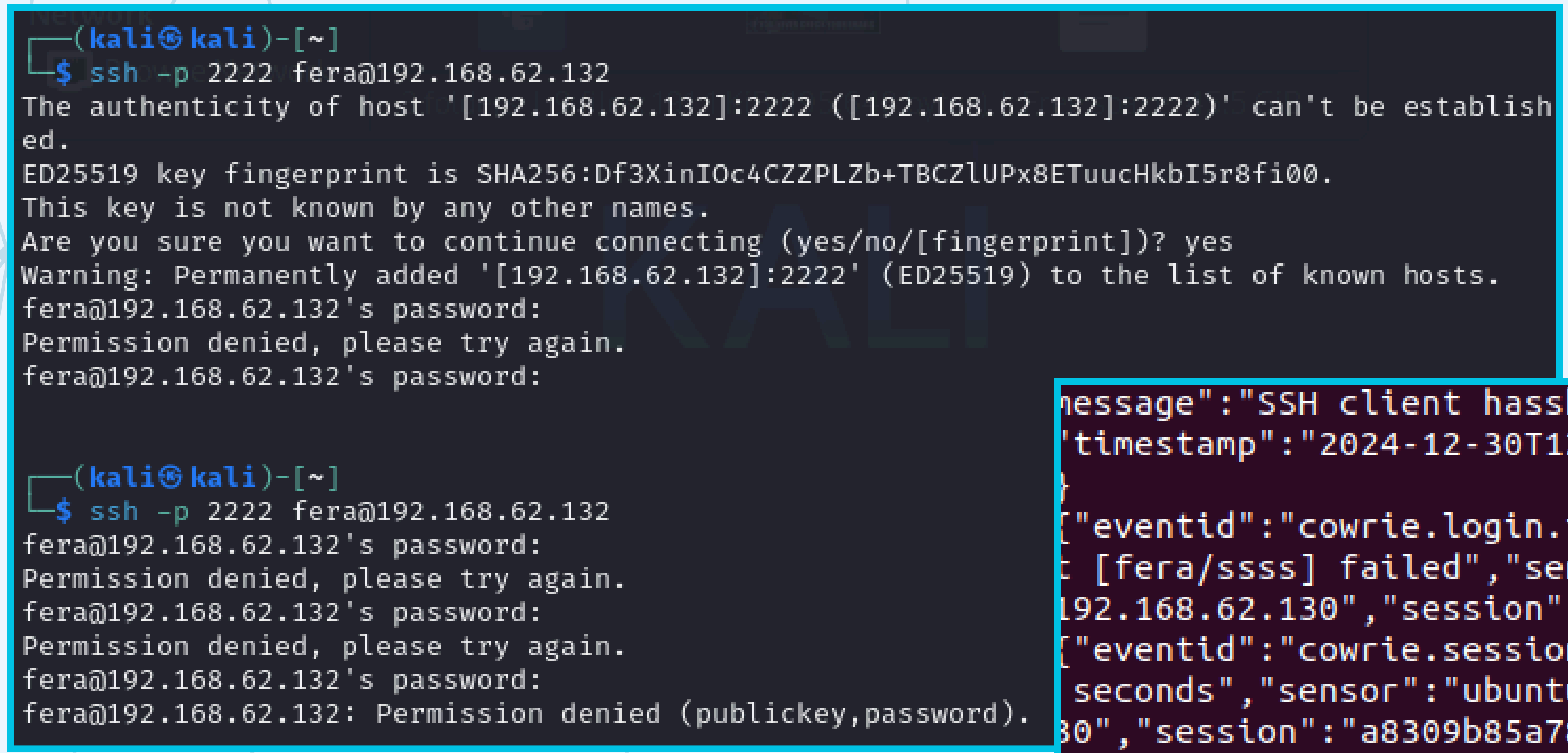


```
  GNU nano 2.9.3                                                  /etc/apparmor.d/opt.dionaea

# Last Modified: Sun Dec 29 13:55:55 2024
#include <tunables/global>

/opt/dionaea {
  #include <abstractions/base>

   # Règles pour les fichiers et répertoires
  /opt/dionaea/** r,
  /opt/dionaea/bin/dionaea px,
  /var/lib/dionaea/** rw,
  /var/log/dionaea/** rw,
  /etc/dionaea/** r,
  /tmp/** rw,

  # Capacités nécessaires
  capability net_bind_service,
  capability sys_admin,
  capability setuid,
  capability setgid,
  capability dac_override,
  capability sys_chroot,

}
```



```
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
```

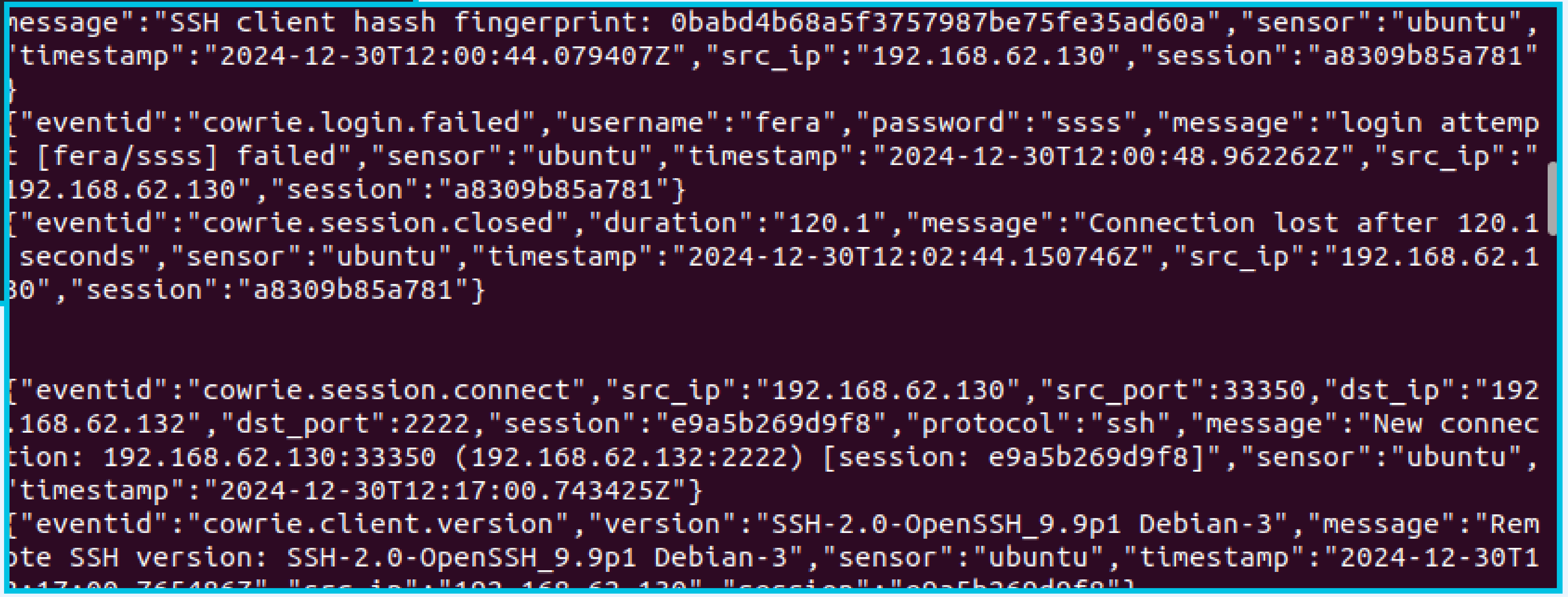
ATTACK SIMULATION



```
┌──(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).
```

SSH Attack

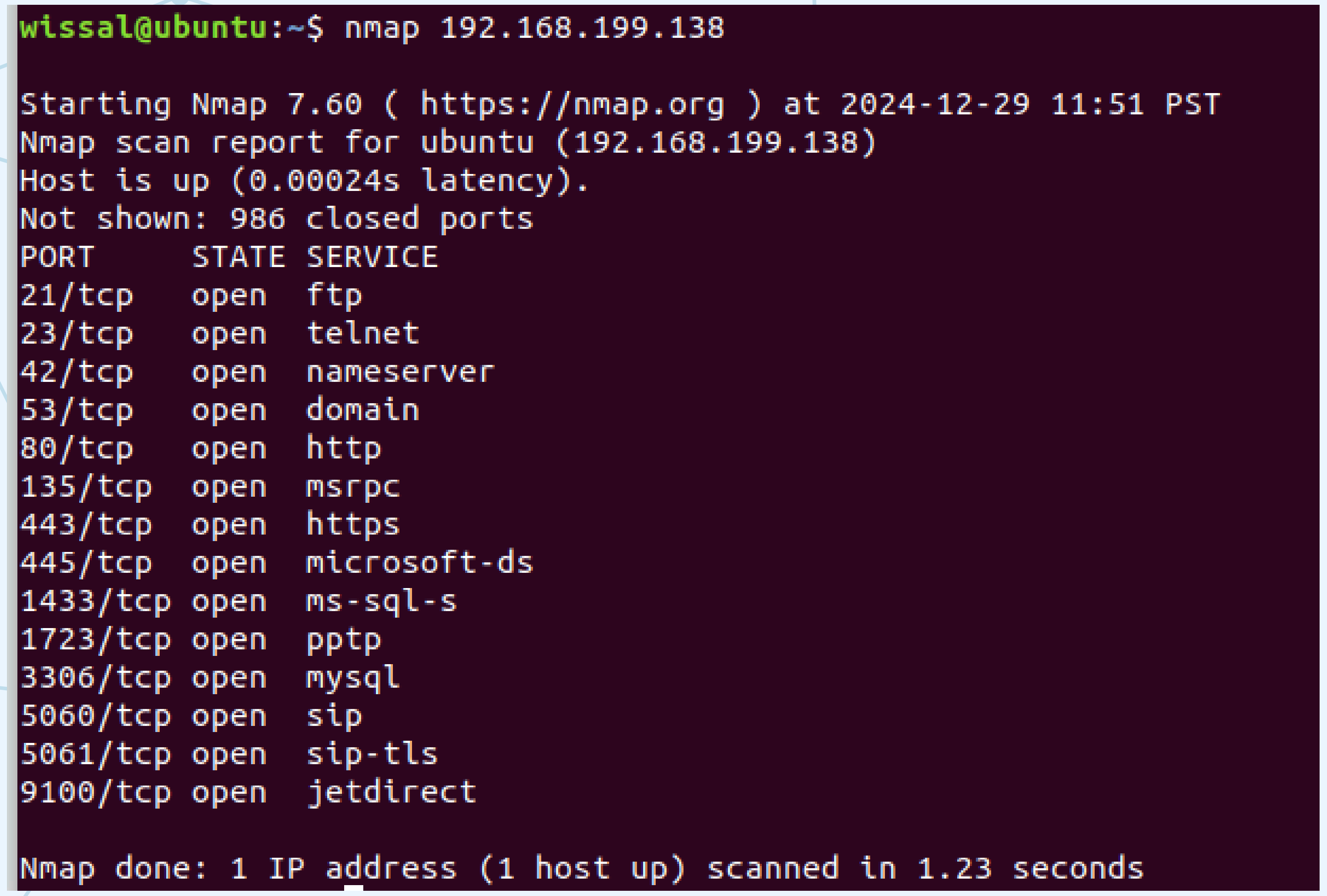
Cowrie's response



```
message":"SSH client hassh fingerprint: 0babd4b68a5f3757987be75fe35ad60a","sensor":"ubuntu",
"timestamp":"2024-12-30T12:00:44.079407Z","src_ip":"192.168.62.130","session":"a8309b85a781"
}
{"eventid":"cowrie.login.failed","username":"fera","password":"ssss","message":"login attemp
t [fera/ssss] failed","sensor":"ubuntu","timestamp":"2024-12-30T12:00:48.962262Z","src_ip":"
192.168.62.130","session":"a8309b85a781"}
{"eventid":"cowrie.session.closed","duration":"120.1","message":"Connection lost after 120.1
 seconds","sensor":"ubuntu","timestamp":"2024-12-30T12:02:44.150746Z","src_ip":"192.168.62.1
30","session":"a8309b85a781"}


{"eventid":"cowrie.session.connect","src_ip":"192.168.62.130","src_port":33350,"dst_ip":"192
.168.62.132","dst_port":2222,"session":"e9a5b269d9f8","protocol":"ssh","message":"New connec
tion: 192.168.62.130:33350 (192.168.62.132:2222) [session: e9a5b269d9f8]","sensor":"ubuntu",
"timestamp":"2024-12-30T12:17:00.743425Z"}
{"eventid":"cowrie.client.version","version":"SSH-2.0-OpenSSH_9.9p1 Debian-3","message":"Rem
ote SSH version: SSH-2.0-OpenSSH_9.9p1 Debian-3","sensor":"ubuntu","timestamp":"2024-12-30T1
2:17:00.765486Z","src_ip":"192.168.62.130","session":"e9a5b269d9f8"}
```

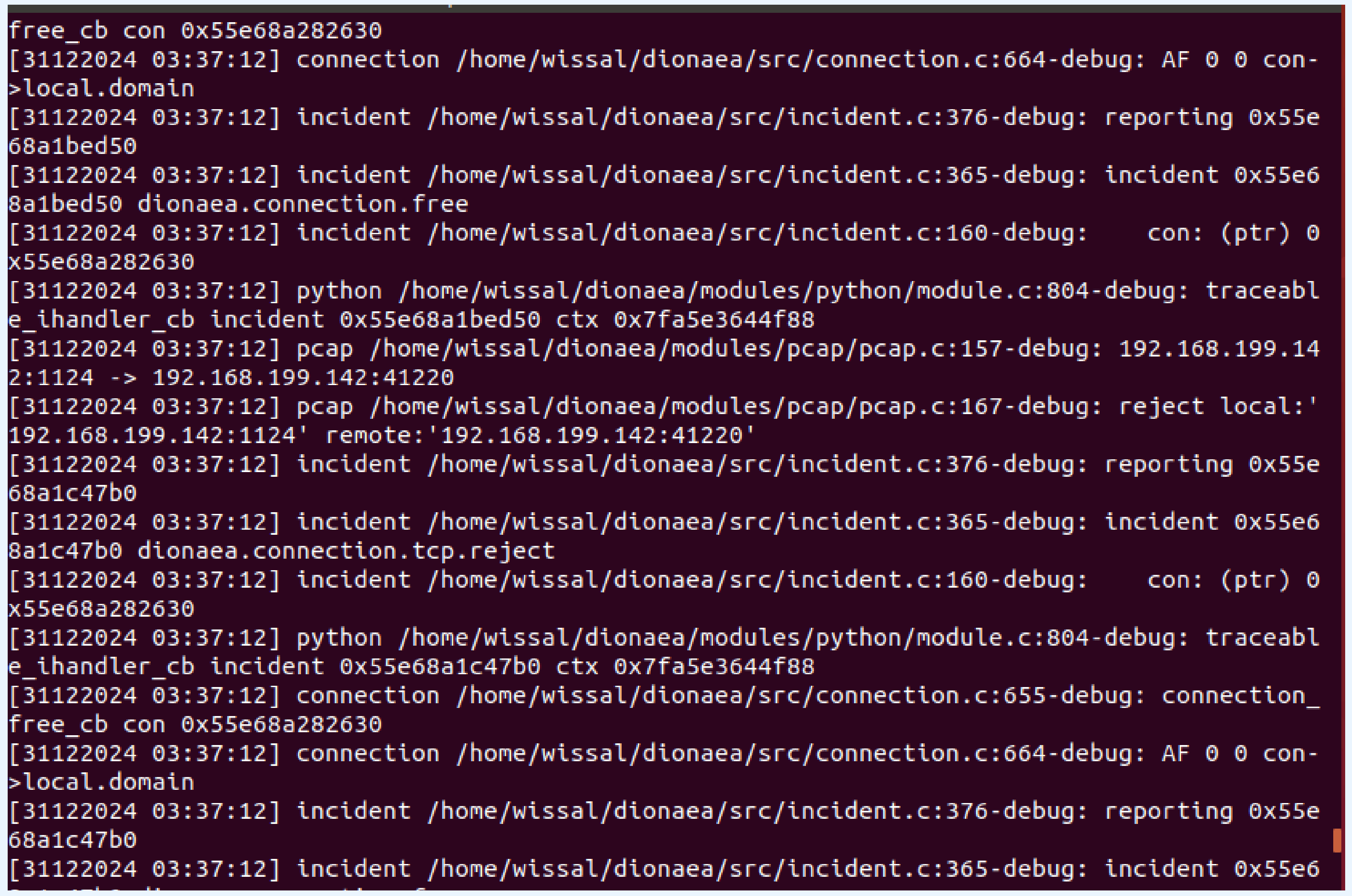
ATTACK SIMULATION

Launching attack



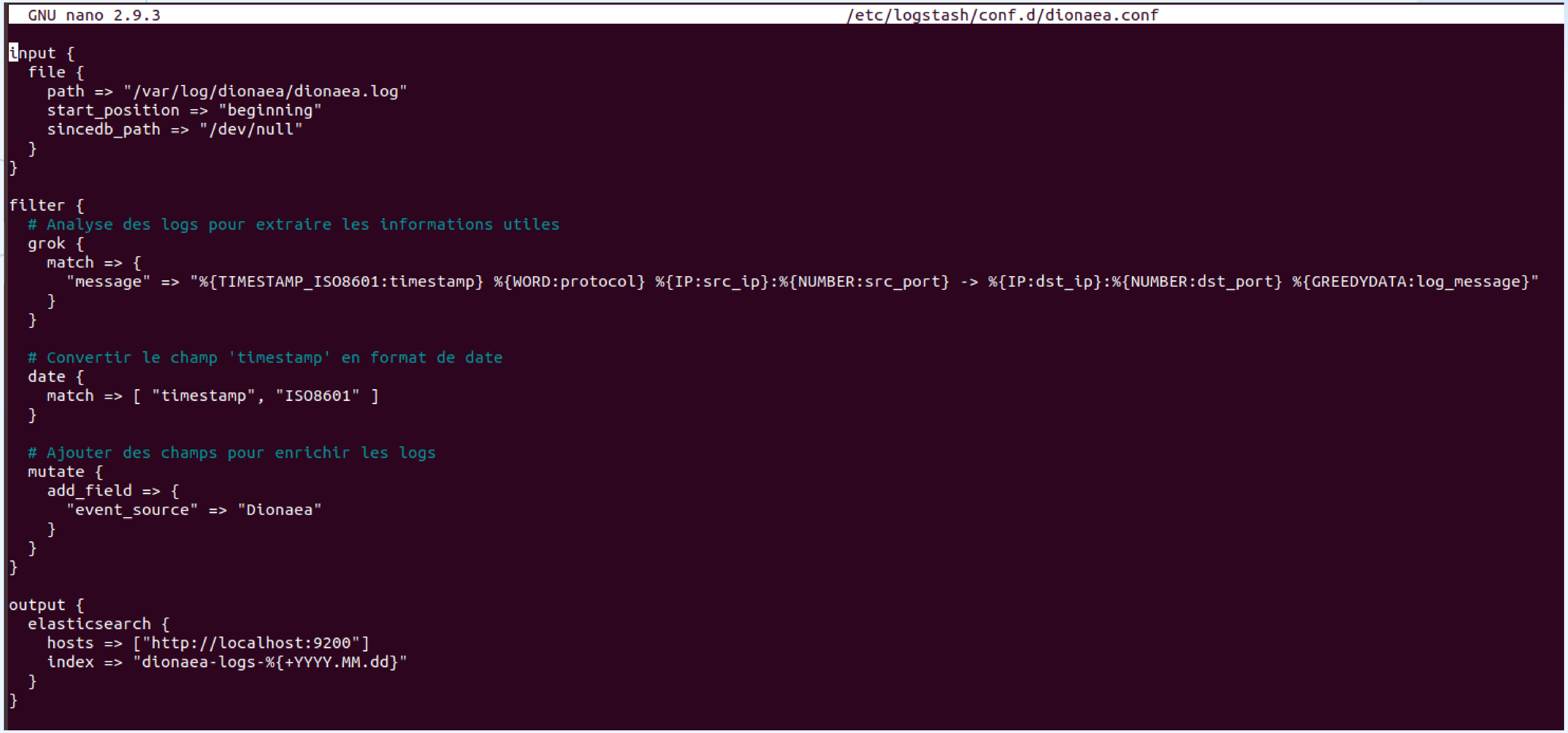
```
wissal@ubuntu:~$ nmap 192.168.199.138

Starting Nmap 7.60 ( https://nmap.org ) at 2024-12-29 11:51 PST
Nmap scan report for ubuntu (192.168.199.138)
Host is up (0.00024s latency).
Not shown: 986 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
23/tcp   open  telnet
42/tcp   open  nameserver
53/tcp   open  domain
80/tcp   open  http
135/tcp  open  msrpc
443/tcp  open  https
445/tcp  open  microsoft-ds
1433/tcp open  ms-sql-s
1723/tcp open  pptp
3306/tcp open  mysql
5060/tcp open  sip
5061/tcp open  sip-tls
9100/tcp open  jetdirect

Nmap done: 1 IP address (1 host up) scanned in 1.23 seconds
```

Dionaea response



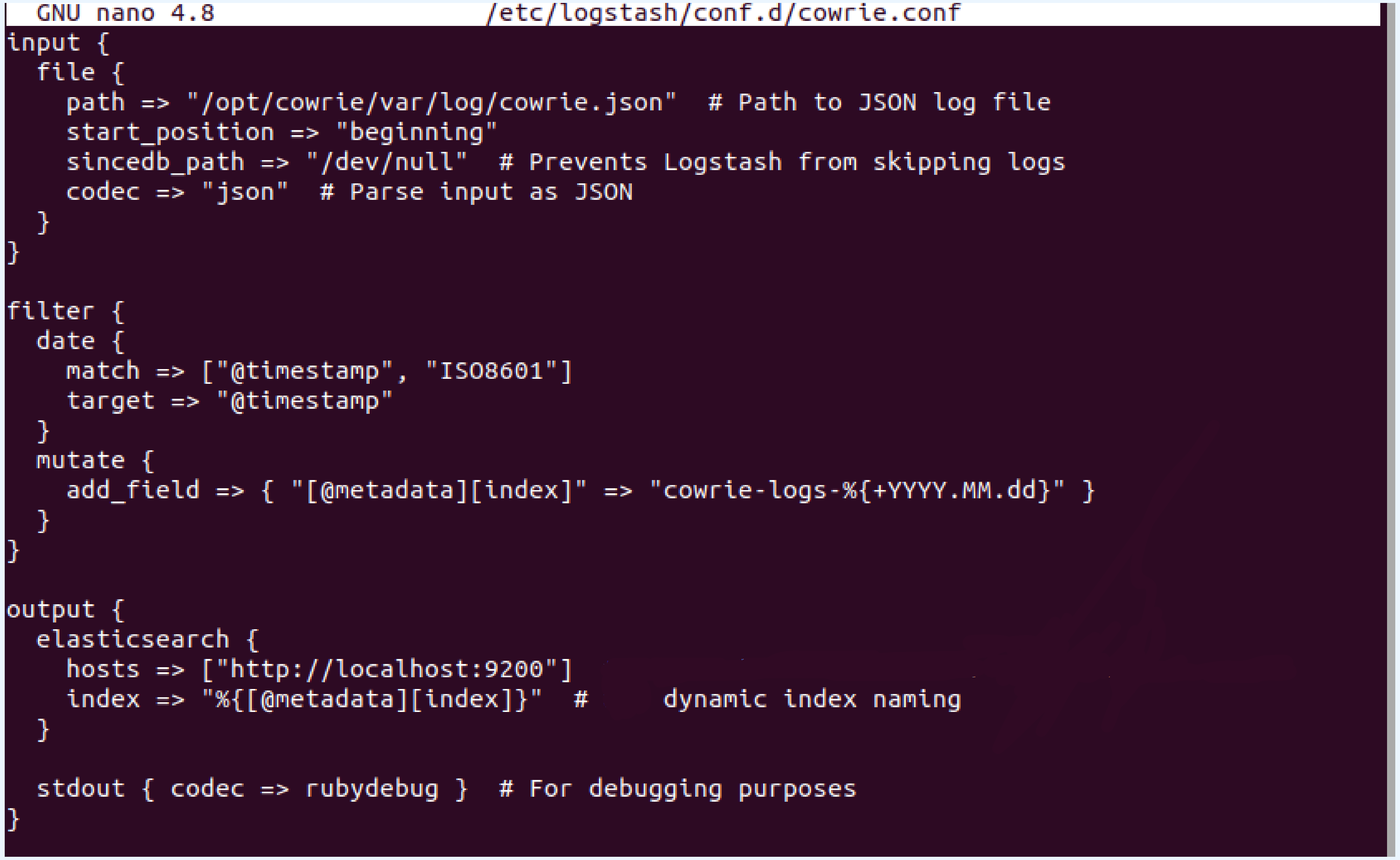
```
free_cb con 0x55e68a282630
[31122024 03:37:12] connection /home/wissal/dionaea/src/connection.c:664-debug: AF 0 0 con-
>local.domain
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:376-debug: reporting 0x55e
68a1bed50
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:365-debug: incident 0x55e6
8a1bed50 dionaea.connection.free
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:160-debug:    con: (ptr) 0
x55e68a282630
[31122024 03:37:12] python /home/wissal/dionaea/modules/python/module.c:804-debug: traceabl
e_ihandler_cb incident 0x55e68a1bed50 ctx 0x7fa5e3644f88
[31122024 03:37:12] pcap /home/wissal/dionaea/modules/pcap/pcap.c:157-debug: 192.168.199.14
2:1124 -> 192.168.199.142:41220
[31122024 03:37:12] pcap /home/wissal/dionaea/modules/pcap/pcap.c:167-debug: reject local:'
192.168.199.142:1124' remote:'192.168.199.142:41220'
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:376-debug: reporting 0x55e
68a1c47b0
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:365-debug: incident 0x55e6
8a1c47b0 dionaea.connection.tcp.reject
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:160-debug:    con: (ptr) 0
x55e68a282630
[31122024 03:37:12] python /home/wissal/dionaea/modules/python/module.c:804-debug: traceabl
e_ihandler_cb incident 0x55e68a1c47b0 ctx 0x7fa5e3644f88
[31122024 03:37:12] connection /home/wissal/dionaea/src/connection.c:655-debug: connection_
free_cb con 0x55e68a282630
[31122024 03:37:12] connection /home/wissal/dionaea/src/connection.c:664-debug: AF 0 0 con-
>local.domain
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:376-debug: reporting 0x55e
68a1c47b0
[31122024 03:37:12] incident /home/wissal/dionaea/src/incident.c:365-debug: incident 0x55e6
```

ELK STACK CONFIGURATION



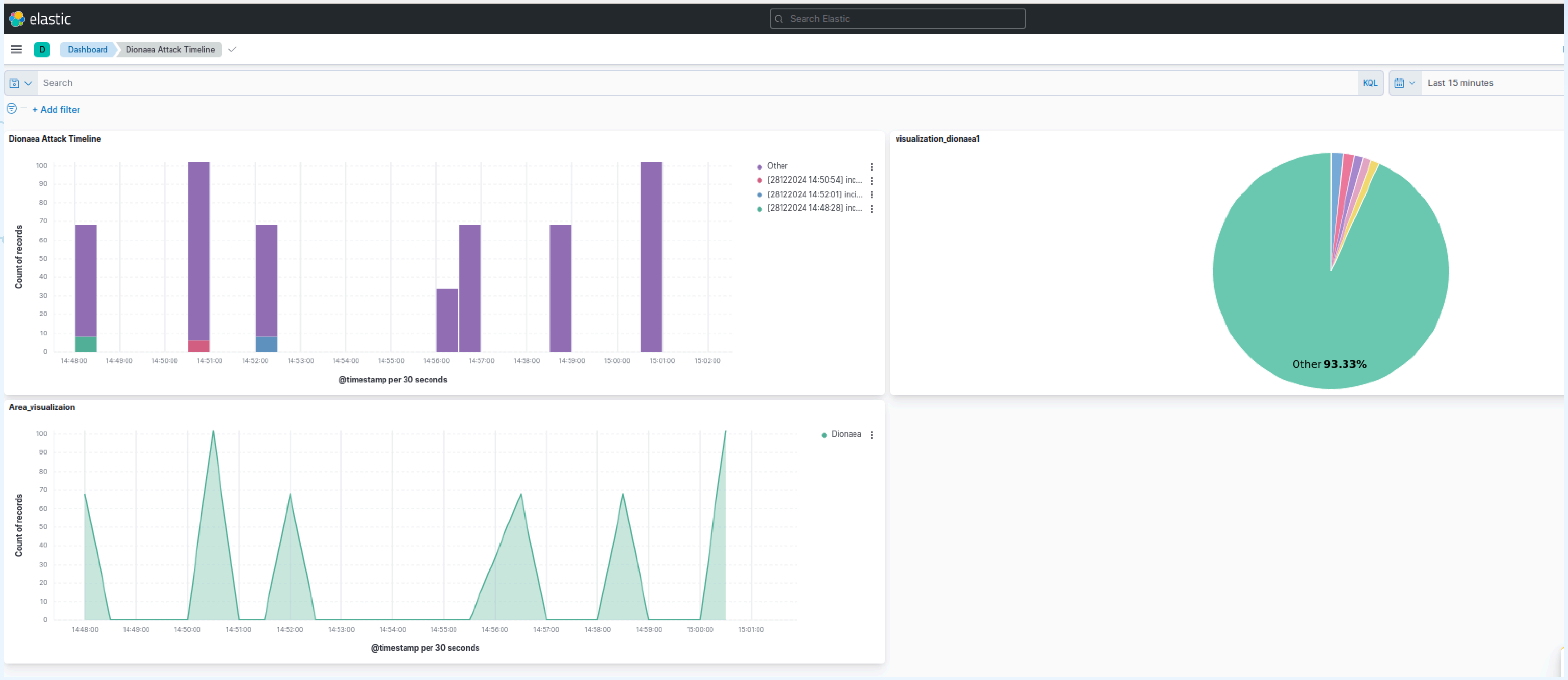
```
  GNU nano 2.9.3                                    /etc/logstash/conf.d/dionaea.conf

input {
  file {
    path => "/var/log/dionaea/dionaea.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  # Analyse des logs pour extraire les informations utiles
  grok {
    match => {
      "message" => "%{TIMESTAMP_ISO8601:timestamp} %{WORD:protocol} %{IP:src_ip}:%{NUMBER:src_port} -> %{IP:dst_ip}:%{NUMBER:dst_port} %{GREEDYDATA:log_message}"
    }
  }

  # Convertir le champ 'timestamp' en format de date
  date {
    match => [ "timestamp", "ISO8601" ]
  }

  # Ajouter des champs pour enrichir les logs
  mutate {
    add_field => {
      "event_source" => "Dionaea"
    }
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "dionaea-logs-%{+YYYY.MM.dd}"
  }
}
```

ELK STACK CONFIGURATION

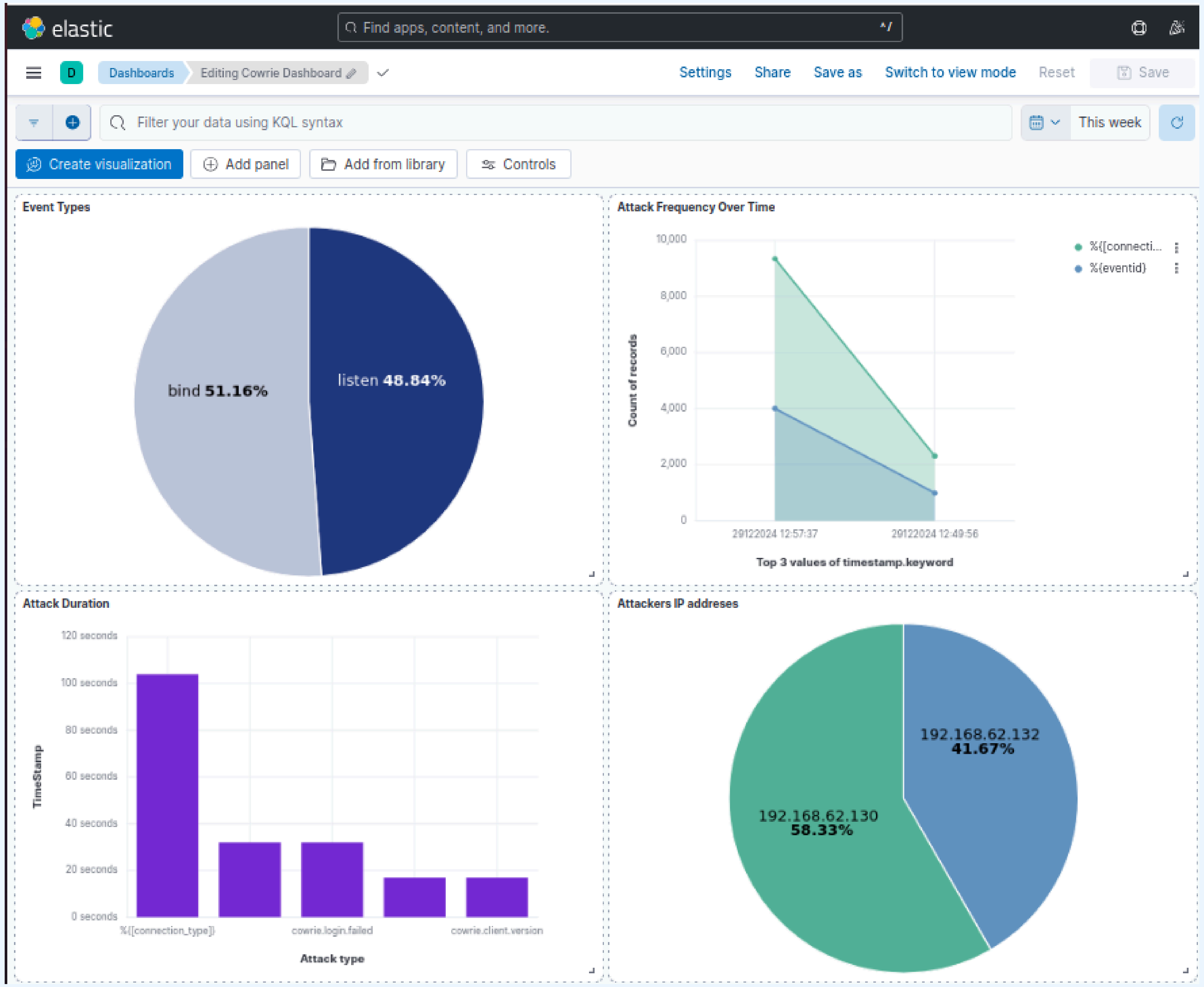


```
  GNU nano 4.8                  /etc/logstash/conf.d/cowrie.conf
input {
  file {
    path => "/opt/cowrie/var/log/cowrie.json"  # Path to JSON log file
    start_position => "beginning"
    sincedb_path => "/dev/null"  # Prevents Logstash from skipping logs
    codec => "json"  # Parse input as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

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

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "%{[@metadata][index]}"  #     dynamic index naming
  }

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

VISUALIZATION



VISUALIZATION



CHALLENGES

- 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.

DEMO

CONCLUSION

This project demonstrated the use of honeypots to understand attacker behavior and gather valuable insights into cybersecurity threats. By deploying Dionaea, Cowrie, and HoneyD, we successfully simulated attack scenarios and analyzed data to identify attack patterns. To keep the honeypots secure, we implemented AppArmor and Seccomp, ensuring they remained protected during operation.
The visualized results provided a clear understanding of how attackers target systems, offering practical insights to improve defenses. While there are areas for improvement, this project highlights the importance of using honeypots as a powerful tool for learning and strengthening cybersecurity measures.

THANK YOU FOR YOUR ATTENTION