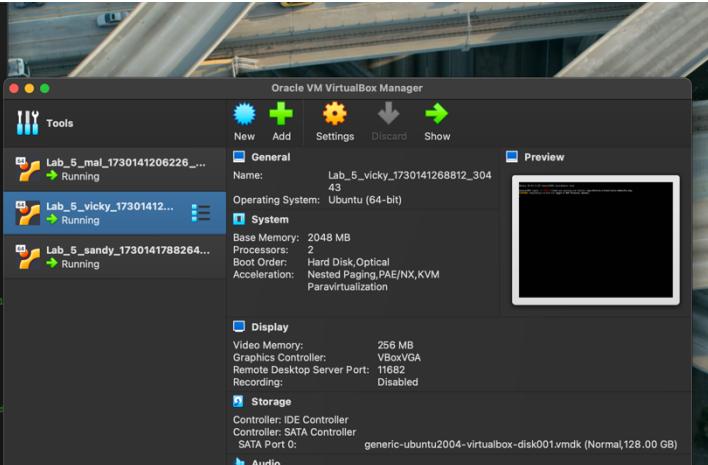
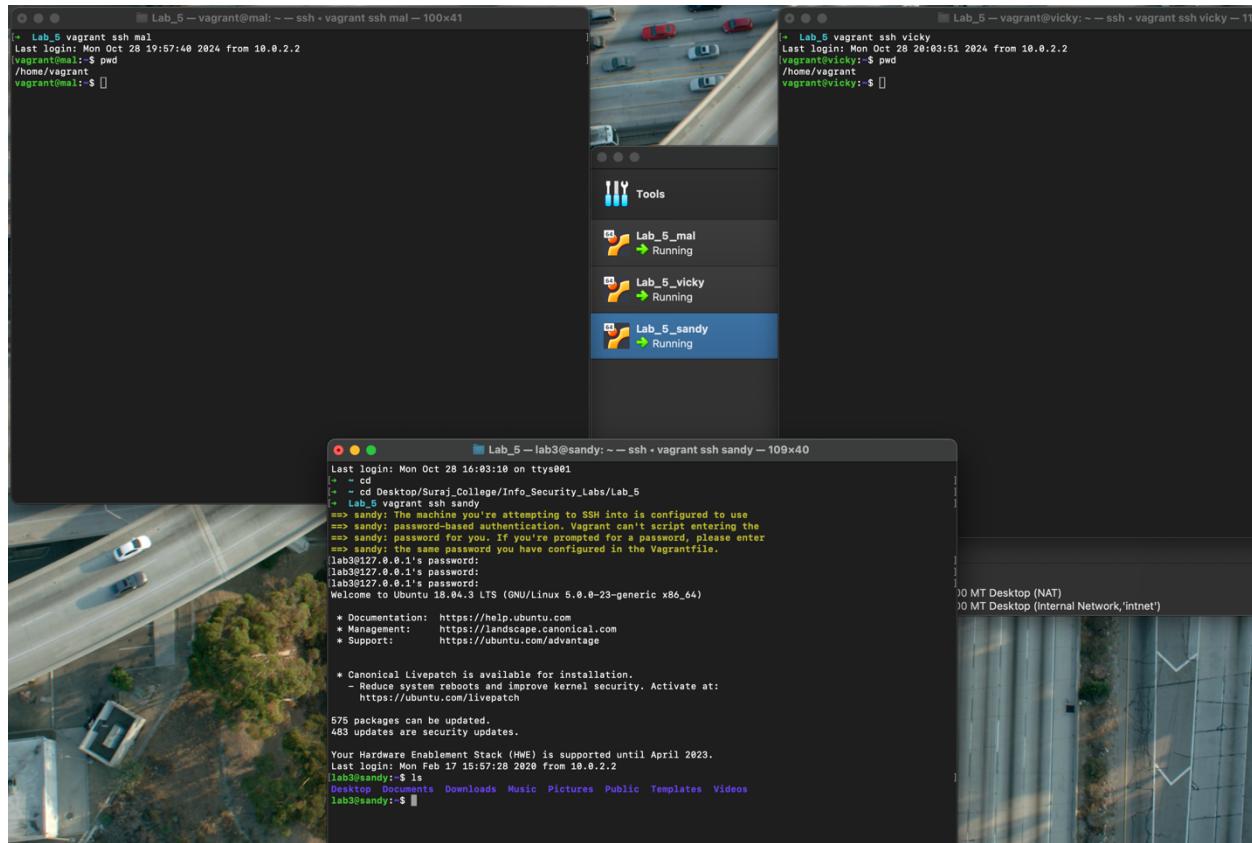


## DIY Network-based IDS: Project 5

First, I initialized the environment with Vagrant file with the command “vagrant up” which creates three distinct headless machines in virtual box.

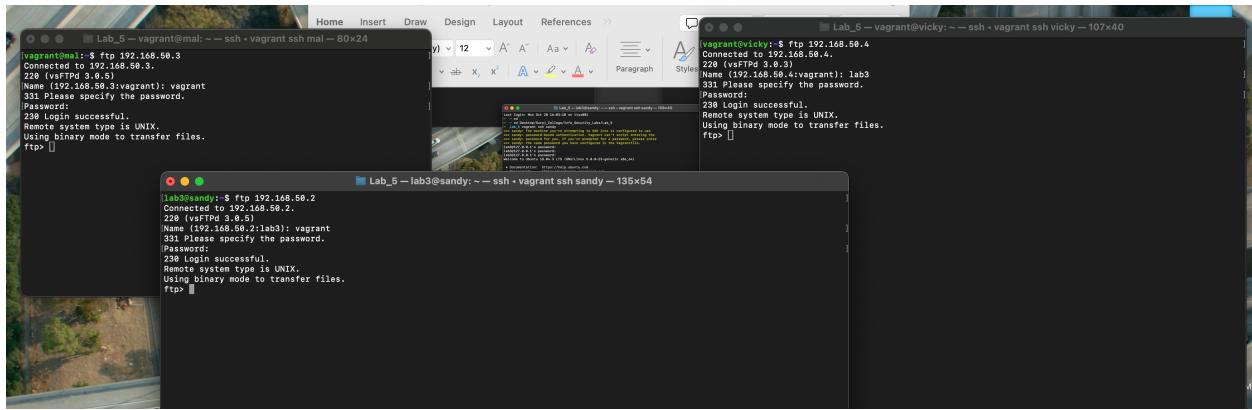


Afterward, I started each machine from my terminal using the command: `vagrant ssh [hostname]`.

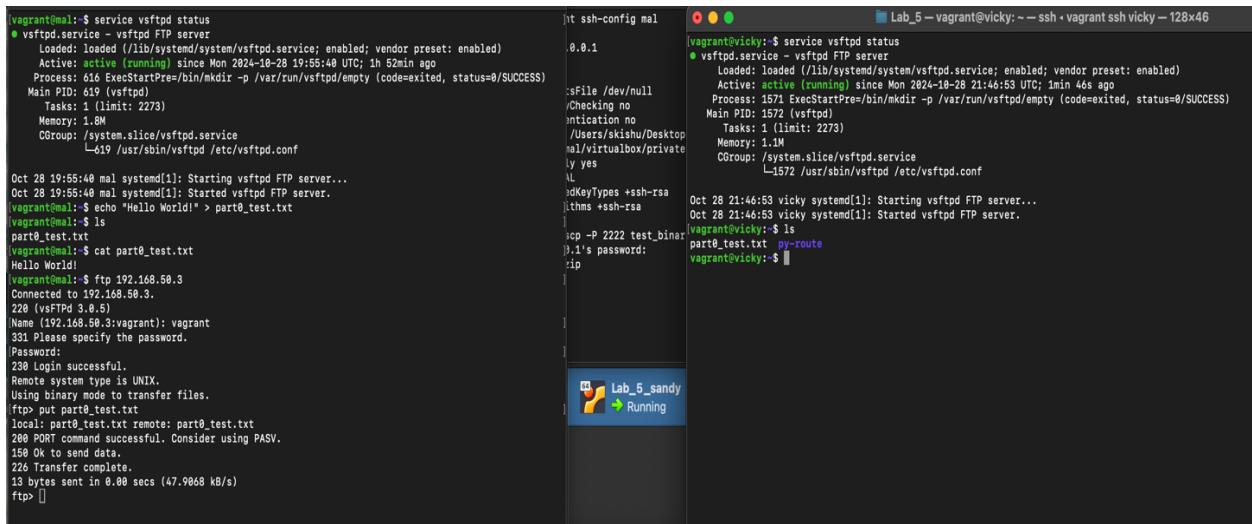


## Part – 0:

In this part, first I verified that FTP services were running on all machines by executing “\$ftp [ip address].



Then, I create a sample text file containing the text “Hello World!” in mal machine. After that I transferred this file to Vicky machine over FTP. First, I started the FTP process with “ftp 192.168.50.3” then entered the required username and password for a successfully login, and sent the file named part0\_test.txt using the command “put part0\_test.txt”. I then confirmed the file transfer on vicky by running the ls command, which showed the file had been successfully received.



## Part – 1:

In this part, I downloaded the test\_binaries.zip file to my host machine, transferred it to the 'mal' machine using scp, and extracted the contents with the unzip command.

```

Lab_5 — vagrant@mail: ~/test_binaries — ssh - vagrant ssh mal — 86x29
[vagrant@mail]:$ ls
part0_test.txt  test_binaries  test_binaries.zip
[vagrant@mail]:$ cd test_binaries/
[vagrant@mail:~/test_binaries]$ ls
embedded_hello-world.pdf  embedded_shell.pdf  shell
embedded_image.pdf        rev_shell           wnidami
[vagrant@mail:~/test_binaries]$ []

Lab_5 — skishu@MacBookPro — ..ty_Labs/Lab_5 — zsh — 80x24
[+ Lab_5 vagrant ssh-config mal
Host mal
HostName 127.0.0.1
User vagrant
Port 2222
UserKnownHostsFile /dev/null
StrictHostKeyChecking no
PasswordAuthentication no
IdentityFile /Users/skishu/Desktop/Suraj_College/Info_Security_Labs/Lab_5/.vagrant/machines/mal/virtualbox/private_key
IdentitiesOnly yes
LogLevel FATAL
PubkeyAcceptedKeyTypes +ssh-rsa
HostKeyAlgorithms +ssh-rsa

[+ Lab_5 sudo scp -P 2222 test_binaries.zip vagrant@127.0.0.1:/home/vagrant/
[vagrant@127.0.0.1's password:
test_binaries.zip                                         100% 8586      3.6MB/s   00:00
+ Lab_5 ]

```

On the Vicky machine, I added a new Snort rule in the /etc/snort/rules directory under the local.rules file to detect executable files transmitted over the network. I also enabled write access on Vicky to avoid permission issues.

```

GNU nano 4.8                               local.rules
var MALWARE_NET [192.168.0.0/16]

# CNC
alert tcp $MALWARE_NET any <> any any \
  (msg:"cnc"; content: "cnc_sig"; \
  sid:10000006; rev:001;)

# Part 1
alert tcp $MALWARE_NET any <> any any \
  (msg:"Executable file detected"; \
  file_data; content: "|7F 45 4C 46|";\
  offset:0; depth:300; \
  metadata:service ftp; \
  sid:10000007; rev:001;)

[ Wrote 15 lines ]
^G Get Help      ^O Write Out    ^W Where Is      ^K Cut Text      ^J Justify      ^C Cur Pos      M-U Undo
^X Exit          ^R Read File     ^\ Replace       ^U Paste Text    ^T To Spell     ^_ Go To Line   M-E Redo

```

Then, I established a Snort console on Vicky using this command:

```
sudo snort -A console -i eth1 -u snort -g snort -c /etc/snort/snort.conf
```

Afterward, I transferred all files from the test\_binaries directory on 'mal' to 'Vicky' via ftp 192.168.50.3. Snort successfully captured log entries for these transfers, displaying them on the console in Vicky. However, I could only capture four files out of the entire transfer.

The screenshot shows two terminal windows. The left window is on host 'mal' (IP 192.168.50.3) and the right window is on host 'vicky' (IP 192.168.50.2). Both windows show Snort logs. The left window shows the transfer of several PDF files (embedded\_shell.pdf, embedded\_hello-world.pdf) via FTP to 'vicky'. The right window shows the corresponding Snort log entries for these file transfers, including details like packet counts and sequence numbers.

```
Lab_5 — vagrant@mal: ~/test_binaries — ssh -v vagrant ssh mal — 86x62
150 Ok to send data.
226 Transfer complete.
745 bytes sent in 0.00 secs (552.8412 kB/s)
ftp> exit
421 Timeout.

vagrant@mal:~/test_binaries$ ftp 192.168.50.3
Connected to 192.168.50.3.
220 (vsFTPd 3.0.5)
Name (192.168.50.3:vagrant): vagrant
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put embedded_shell.pdf
local: embedded_shell.pdf remote: embedded_shell.pdf
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
745 bytes sent in 0.00 secs (2.8765 MB/s)
ftp> exit
421 Timeout.

vagrant@mal:~/test_binaries$ ftp 192.168.50.3
Connected to 192.168.50.3.
220 (vsFTPd 3.0.5)
Name (192.168.50.3:vagrant): vagrant
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put whoami
local: whoami remote: whoami
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
166 bytes sent in 0.00 secs (757.5204 kB/s)
ftp> put rev_shell
local: rev_shell remote: rev_shell
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
194 bytes sent in 0.00 secs (633.6225 kB/s)
ftp> put embedded_hello-world.pdf
local: shell remote: shell
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
162 bytes sent in 0.00 secs (632.8125 kB/s)
ftp> put embedded_shell.pdf
local: embedded_shell.pdf remote: embedded_shell.pdf
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
745 bytes sent in 0.00 secs (2.8774 MB/s)
ftp> put embedded_hello-world.pdf
local: embedded_hello-world.pdf remote: embedded_hello-world.pdf
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
16632 bytes sent in 0.00 secs (11.3216 MB/s)
ftp>

Lab_5 — vagrant@vicky: ~ — ssh -v vagrant ssh vicky — 107x56
[...]
| Match States : 556
| Memory (MB) : 2.91
| Patterns : 6.06
| Match Lists : 0.11
| DFA :
|   1 byte states : 2.27
|   2 byte states : 2.42
|   4 byte states : 0.08
| [ Number of patterns truncated to 20 bytes: 15 ]
pcap DAQ configured to passive.
Activating network traffic from "eth1".
[...]
| Reloading rules starting...
|> Reload thread started, thread 0x7f728a4a6700 (1375)
|> Decoding Ethernet
|> Set pid to 1081
|> Set uid to 998
|> == Initialization Complete ==
|> Snort! <-
|> Version 2.9.20 GRe (Build 82)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.9.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11
Rules Engine: SF_SNORT_DETECTION_ENGIN Version 3.2 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_SSHV1 Version 1.0 <Build 1>
Preprocessor Object: SF_SSHV2 Version 1.1 <Build 4>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_SDP Version 1.1 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_STOMPPLUS Version 1.0 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: apid Version 1.1 <Build 5>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 3>
Preprocessor Object: SF_GTPV1 Version 1.1 <Build 3>
Preprocessor Object: SF_GTPV2 Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SIP Version 1.1 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Commencing packet processing (pid=1374)
11/05/00:15:39.007522 [*] [1:10000007:1] Executable file detected [**] [Priority: 0] {TCP} 192.168.50.2:3909 -> 192.168.50.3:20
11/05/00:15:43.859990 [*] [1:10000007:1] Executable file detected [**] [Priority: 0] {TCP} 192.168.50.2:6311 -> 192.168.50.3:20
11/05/00:15:49.076516 [*] [1:10000007:1] Executable file detected [**] [Priority: 0] {TCP} 192.168.50.2:6963 -> 192.168.50.3:20
11/05/00:15:55.243862 [*] [1:10000007:1] Executable file detected [**] [Priority: 0] {TCP} 192.168.50.2:6931 -> 192.168.50.3:20
02728115:1002728277, ack 25752498
2
57416912:2457416206, ack 33218241
4
05337467:4088337633, ack 15974888
```

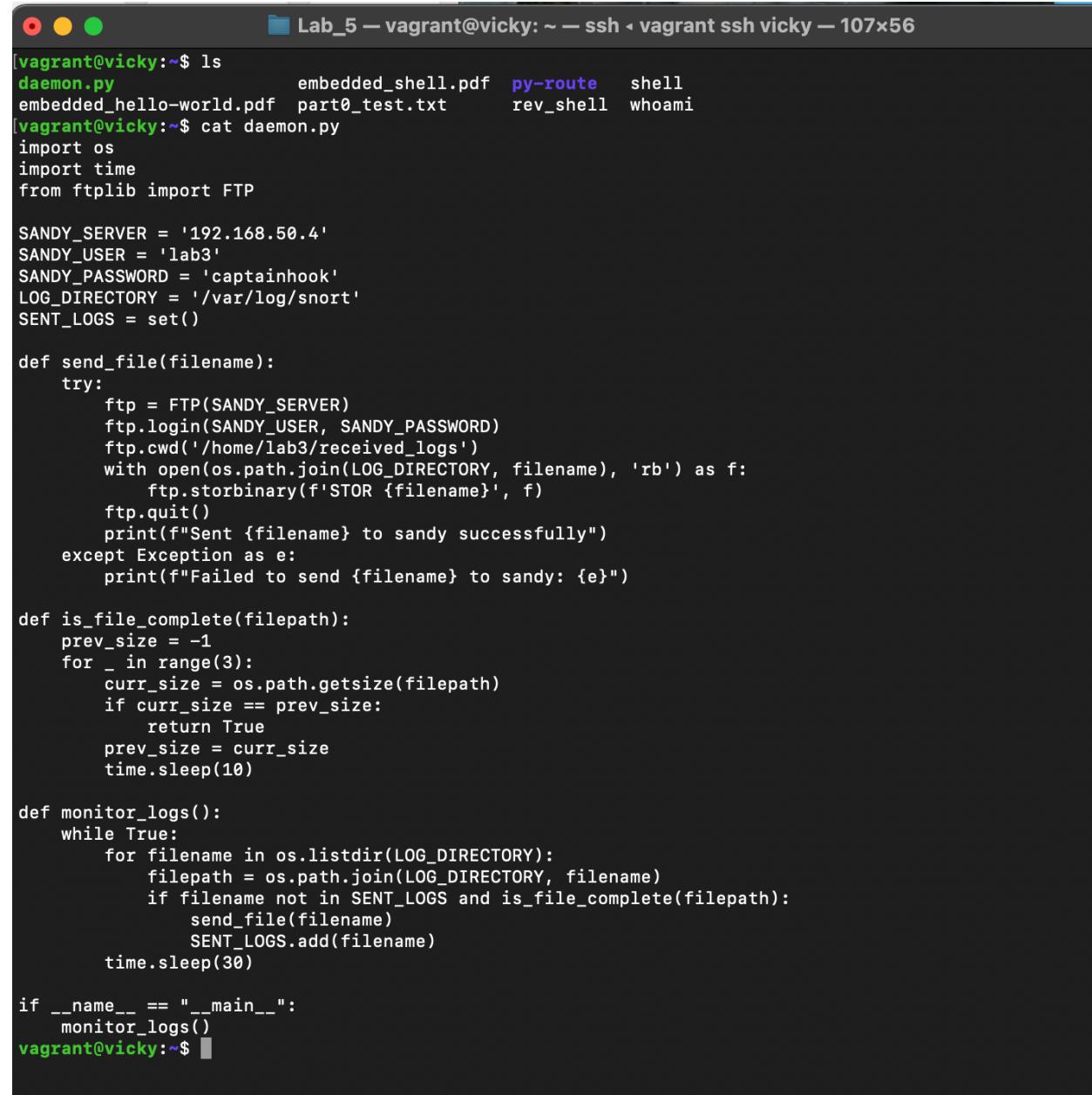
Snort saved all the log of file transfer and save in the file in /var/log/snort directory.

The screenshot shows a terminal window on host 'vicky' displaying the contents of the /var/log/snort directory. It lists several log files, notably 'snort.log.1730765734' and 'snort.log.1730765734'. The user then runs 'tcpdump -r snort.log.1730765734' to analyze the captured traffic.

```
vagrant@vicky:~$ cd /var/log/snort
vagrant@vicky:/var/log/snort$ ls
snort.log.1730765734
vagrant@vicky:/var/log/snort$ sudo tcpdump -r snort.log.1730765734
reading from file snort.log.1730765734, link-type EN10MB (Ethernet)
00:15:39.007522 IP 192.168.50.2.53909 > vicky.ftp-data: Flags [P.], seq 3320709849:3320710015, ack 24165251
38, win 510, options [nop,nop,TS val 1637239106 ecr 3748408245], length 166
00:15:43.859990 IP 192.168.50.2.50311 > vicky.ftp-data: Flags [P.], seq 2637153977:2637154171, ack 95495355
6, win 510, options [nop,nop,TS val 1637243966 ecr 3748413097], length 194
00:15:49.076516 IP 192.168.50.2.60963 > vicky.ftp-data: Flags [P.], seq 4098968854:4098969016, ack 18249539
20, win 510, options [nop,nop,TS val 1637249191 ecr 3748418314], length 162
00:15:55.243862 IP 192.168.50.2.60131 > vicky.ftp-data: Flags [P.], seq 2936944336:2936945081, ack 68535096
9, win 510, options [nop,nop,TS val 1637255368 ecr 3748424481], length 745
vagrant@vicky:/var/log/snort$
```

## Part – 2:

In this part, I created a Python daemon script named `daemon.py` to automate the process of sending Snort log files from the `/var/log/snort` directory on Vicky to 'Sandy' whenever a new log file is generated.



```
[vagrant@vicky:~$ ls
daemon.py          embedded_shell.pdf  py-route  shell
embedded_hello-world.pdf  part0_test.txt      rev_shell  whoami
[vagrant@vicky:~$ cat daemon.py
import os
import time
from ftplib import FTP

SANDY_SERVER = '192.168.50.4'
SANDY_USER = 'lab3'
SANDY_PASSWORD = 'captainhook'
LOG_DIRECTORY = '/var/log/snort'
SENT_LOGS = set()

def send_file(filename):
    try:
        ftp = FTP(SANDY_SERVER)
        ftp.login(SANDY_USER, SANDY_PASSWORD)
        ftp.cwd('/home/lab3/received_logs')
        with open(os.path.join(LOG_DIRECTORY, filename), 'rb') as f:
            ftp.storbinary(f'STOR {filename}', f)
        ftp.quit()
        print(f"Sent {filename} to sandy successfully")
    except Exception as e:
        print(f"Failed to send {filename} to sandy: {e}")

def is_file_complete(filepath):
    prev_size = -1
    for _ in range(3):
        curr_size = os.path.getsize(filepath)
        if curr_size == prev_size:
            return True
        prev_size = curr_size
    time.sleep(10)

def monitor_logs():
    while True:
        for filename in os.listdir(LOG_DIRECTORY):
            filepath = os.path.join(LOG_DIRECTORY, filename)
            if filename not in SENT_LOGS and is_file_complete(filepath):
                send_file(filename)
                SENT_LOGS.add(filename)
    time.sleep(30)

if __name__ == "__main__":
    monitor_logs()
vagrant@vicky:~$
```

Additionally, I create a `daemon.service` file in `/etc/systemd/system` directory to run `daemon.py` in the background automatically.

```

● ● ● Lab_5 — vagrant@vicky: /etc/systemd/system — ssh - vagrant ssh vicky — 107x56
[vagrant@vicky:~$ cd /etc/systemd/system/
[vagrant@vicky:/etc/systemd/system$ ls
cloud-final.service.wants      getty.target.wants      rescue.target.wants
daemon.service                  graphical.target.wants sleep.target.wants
dbus-org.freedesktop.ModemManager1.service iscsi.service    sockets.target.wants
dbus-org.freedesktop.network1.service mdmonitor.service   sshd.service
dbus-org.freedesktop.resolve1.service multipath-tools.service sysinit.target.wants
dbus-org.freedesktop.timesync1.service multi-user.target.wants syslog.service
default.target.wants            network-online.target.wants timers.target.wants
emergency.target.wants          open-vm-tools.service.requires vmtoolsd.service
final.target.wants              paths.target.wants
[vagrant@vicky:/etc/systemd/system$ sudo cat daemon.service
[Unit]
Description=Daemon to send Snort logs from vicky to sandy over FTP
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/vagrant/daemon.py
Restart=always
User=root

[Install]
WantedBy=multi-user.target
[vagrant@vicky:/etc/systemd/system$ ]

```

With this setup, whenever a file is sent from 'mal' to Vicky, the daemon on Vicky detects the new Snort log file and transfers it to the 'Sandy' machine.

```

● ● ● Lab_5 — lab3@sandy: ~/received_logs — ssh - vagrant ssh sandy — 121x50
lab3@sandy:~/received_logs$ ls
snort.log.1730765734
lab3@sandy:~/received_logs$ sudo tcpdump -r snort.log.1730765734
reading from file snort.log.1730765734, link-type EN10MB (Ethernet)
19:15:39.007522 IP 192.168.50.2.53909 > 192.168.50.3.ftp-data: Flags [P.], seq 3320709849:3320710015, ack 2416525138, win 510, options [nop,nop,TS val 1637239106 ecr 3748408245], length 166
19:15:43.859990 IP 192.168.50.2.50311 > 192.168.50.3.ftp-data: Flags [P.], seq 2637153977:2637154171, ack 954953556, win 510, options [nop,nop,TS val 1637243966 ecr 3748413097], length 194
19:15:49.076516 IP 192.168.50.2.60963 > 192.168.50.3.ftp-data: Flags [P.], seq 4098968854:4098969016, ack 1824953920, win 510, options [nop,nop,TS val 1637249191 ecr 3748418314], length 162
19:15:55.243862 IP 192.168.50.2.60131 > 192.168.50.3.ftp-data: Flags [P.], seq 2936944336:2936945081, ack 685350969, win 510, options [nop,nop,TS val 1637255368 ecr 3748424481], length 745
lab3@sandy:~/received_logs$ 

```

### Part - 3: 3a

In this part, first I downloaded the provided hooks.c file, replaced the existing one, and modified the LKM (Loadable Kernel Module) code to log additional details about the syscalls.

```

Lab_5 — lab3@sandy: ~/Desktop/lab_files/src — ssh -v vagrant ssh sandy — 143x68
asmlinkage long new_brk(const struct pt_regs *regs) {
    printk("[+] brk called");
    return original_brk(regs);
}

asmlinkage long new_mmap(const struct pt_regs *regs) {
    printk("[+] mmap called");
    return original_mmap(regs);
}

asmlinkage long new_munmap(const struct pt_regs *regs) {
    printk("[+] munmap called");
    return original_munmap(regs);
}

asmlinkage long new_mprotect(const struct pt_regs *regs) {
    printk(KERN_ALERT "[ALERT] Malicious syscall: mprotect called\n");
    return original_mprotect(regs);
}

asmlinkage long new_clone(const struct pt_regs *regs) {
    printk("[+] clone called");
    return original_clone(regs);
}

asmlinkage long new_fork(const struct pt_regs *regs) {
    printk("[+] fork called");
    return original_fork(regs);
}

asmlinkage long new_execve(const struct pt_regs *regs) {
    char filename[MAX_PATH_LEN];
    long copied = strncpy_from_user(filename, (char __user *)regs->di, sizeof(filename));
    if (copied > 0) {
        printk(KERN_ALERT "[ALERT] Malicious syscall: execve on file: %s\n", filename);
    }
    return original_execve(regs);
}

asmlinkage long new_exit(const struct pt_regs *regs) {
    printk("[+] exit called");
    return original_exit(regs);
}

asmlinkage long new_exit_group(const struct pt_regs *regs) {
    printk("[+] exit_group called");
    return original_exit_group(regs);
}

asmlinkage long new_read(const struct pt_regs *regs) {
    printk("[+] read called");
    return original_read(regs);
}

asmlinkage long new_write(const struct pt_regs *regs) {
    char buf[64];
    unsigned long count = regs->dx;
    unsigned long copied = (count < sizeof(buf)) ? count : sizeof(buf) - 1;

    if (copy_from_user(buf, (char __user *)regs->si, copied) == 0) {
        buf[copied] = '\0';
        printk(KERN_ALERT "[ALERT] Malicious syscall: write to fd %ld, content: %s\n", regs->di, buf);
    }
    return original_write(regs);
}

asmlinkage long new_fstat(const struct pt_regs *regs) {
    printk("[+] fstat called");
}

```

I compiled the code by running make all in the src folder, which created the required files, and then I inserted the module using:

```

Lab_5 — lab3@sandy: ~/Desktop/lab_files/src — ssh -v vagrant ssh sandy — 176x65
[lab3@sandy:~/Desktop/lab_files/src$ ls
hooks.c hooks.h hooks.ko hooks.mod.c hooks.mod.o hooks.o Makefile modules.order Module.symvers
[lab3@sandy:~/Desktop/lab_files/src$ cat Makefile
obj-m += hooks.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
lab3@sandy:~/Desktop/lab_files/src$ 

```

### Part - 3: 3b

For this part, on 'Sandy,' I created another Python daemon script named `sandy_daemon.py`, which monitors new log files transferred from Vicky. This script scans the log file, extracts any executable binaries, and saves them in the `/home/lab3/received_logs` directory.

```
Lab_5 — lab3@sandy: ~ — ssh - vagrant ssh sandy — 121x66
[lab3@sandy:~/received_logs$ cd
[lab3@sandy:~$ cat sandy_daemon.py
import os
import time
import subprocess
from pyshark import FileCapture
import hashlib

LOG_DIR = "/home/lab3/received_logs"
PROCESSED_LOGS = set()

def process_log(log_path):
    if extract_binary(log_path):
        print(f"Finished processing snort log: {log_path}")

def extract_binary(log_path):
    try:
        cap = FileCapture(log_path)
        extracted_binaries = {}

        for packet in cap:
            if hasattr(packet, 'tcp') and hasattr(packet.tcp, 'payload'):
                payload = packet.tcp.payload.binary_value

                elf_index = payload.find(b'\x7fELF')
                if elf_index != -1:
                    elf_payload = payload[elf_index:]
                    payload_hash = hashlib.md5(elf_payload).hexdigest()

                    if payload_hash not in extracted_binaries:
                        exec_path = os.path.join(LOG_DIR, f"exec_binary_{payload_hash}")
                        with open(exec_path, "wb") as f:
                            f.write(elf_payload)
                        extracted_binaries[payload_hash] = exec_path
                        run_binary(exec_path)
                    else:
                        print("Duplicate payload detected.")
                else:
                    print("Binary did not find in this payload")
    except Exception as e:
        print(f"Failed to process snort log {log_path}: {e}")
        return None

    return True if extracted_binaries else None
except Exception as e:
    print(f"Failed to process snort log {log_path}: {e}")
    return None

def run_binary(exec_path):
    os.chmod(exec_path, 0o755)
    subprocess.Popen([exec_path], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

def monitor_logs():
    while True:
        snort_logs = [f for f in os.listdir(LOG_DIR) if f.startswith('snort.log.')]
        for filename in snort_logs:
            file_path = os.path.join(LOG_DIR, filename)
            if filename not in PROCESSED_LOGS:
                process_log(file_path)
                PROCESSED_LOGS.add(filename)
            time.sleep(10)

if __name__ == "__main__":
    monitor_logs()

lab3@sandy:~$
```

Also, I created a systemd service, `sandy_daemon.service`, in `/etc/systemd/system` to run `sandy_daemon.py` in the background continuously.

```
[lab3@sandy:~$ cd /etc/systemd/system/
[lab3@sandy:/etc/systemd/system$ ls
bluetooth.target.wants
cloud-final.service.wants
dbus-fi.w1.wpa_supplicant1.service
dbus-org.bluez.service
dbus-org.freedesktop.Avahi.service
dbus-org.freedesktop.ModemManager1.service
dbus-org.freedesktop.nm-dispatcher.service
dbus-org.freedesktop.resolve1.service
dbus-org.freedesktop.thermald.service
default.target.wants
display-manager.service
display-manager.service.wants
final.target.wants
getty.target.wants
graphical.target.wants
multi-user.target.wants
network-online.target.wants
oem-config.service.wants
paths.target.wants
printer.target.wants
sandy_daemon.service
snap-bare-5.mount
snap-core-17200.mount
snap-core18-1668.mount
snap-core18-2846.mount
snap-core22-1663.mount
snapd.mounts.target.wants
'snap-gnome\x2d3\x2d28\x2d1804-116.mount'
'snap-gnome\x2d3\x2d28\x2d1804-198.mount'
'snap-gnome\x2d42\x2d2204-176.mount'
'snap-gnome\x2dcalculator-544.mount'
'snap-gnome\x2dcalculator-955.mount'
'snap-gnome\x2dcharacters-399.mount'
'snap-gnome\x2dcharacters-797.mount'
'snap-gnome\x2dlogs-123.mount'
'snap-gnome\x2dlogs-81.mount'
'snap-gnome\x2dsystem\x2dmonitor-127.mount'
'snap-gnome\x2dsystem\x2dmonitor-186.mount'
'snap-gtk\x2dcommon\x2dthemes-1440.mount'
'snap-gtk\x2dcommon\x2dthemes-1535.mount'
sockets.target.wants
spice-vdagentd.target.wants
sshd.service
sysinit.target.wants
syslog.service
timers.target.wants
[lab3@sandy:/etc/systemd/system$ sudo cat sandy_daemon.service
[Unit]
Description=Sandbox Daemon for Executable Processing

[Service]
ExecStart=/usr/bin/python3 /home/lab3/sandy_daemon.py
Restart=always
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target

[lab3@sandy:/etc/systemd/system$ ]
```

so here sandy\_daemon is running in background whenever it detects a new log file in /home/lab3/received\_logs folder. It scans the log file and automatically extracts and save binaries in same folder where snort log located as well as run the binary.

```
Lab_5 — lab3@sandy: ~/received_logs — ssh -vagrant ssh sandy — 121x66
[lab3@sandy:~$ ls
binary_files_from_daemon  Documents  Music          Pictures  received_logs  Templates
Desktop                  Downloads   part0_test.txt  Public    sandy_daemon.py  Videos
[lab3@sandy:~$ cd received_logs/
[lab3@sandy:~/received_logs$ ls
exec_binary_2ec992d899d4c05d9bdf670a611154c9  exec_binary_9f378a9a123f64cb6b58b99ed1ff179e  snort.log.1730765734
exec_binary_88e3a99c79c0259f60b2fc9c16dc9ef4  exec_binary_b81d4c94829bdae32f88f2cd94bdd7ed
[lab3@sandy:~/received_logs$ ]
```

After running binaries through the daemon, malicious syscalls log details are capture in /var/log.kern.log file.

```

Nov 5 20:49:57 sandy kernel: [ 1183.596540] Hello world!
Nov 5 20:49:57 sandy kernel: [ 1183.600947] Syscall table address: 00000000ff1de9cc
Nov 5 20:49:57 sandy kernel: [ 1183.600948] sizeof(unsigned long long *): 8
Nov 5 20:49:57 sandy kernel: [ 1183.600948] sizeof(sys_call_table) : 8
Nov 5 20:49:57 sandy kernel: [ 1183.600981] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.600991] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601034] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601039] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601054] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601058] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601071] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601086] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601118] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601125] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601145] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601160] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601183] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601189] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601202] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601205] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601217] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601220] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601231] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601234] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601246] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601249] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601264] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601268] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601279] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601282] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601293] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601297] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601308] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601311] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601323] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601326] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601337] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601341] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601352] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601355] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601367] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601370] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601381] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601384] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601395] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601398] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601409] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601412] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601421] [+] read called
Nov 5 20:49:57 sandy kernel: [ 1183.601424] [+] openat called
Nov 5 20:49:57 sandy kernel: [ 1183.601427] [+] read called

```

## Part – 4

In this part, I created a new conf file in /etc/rsyslog.d and add configuration that capture the redirect messages from LKM containing "[ALERT]". Finally, saved it into the ids\_alerts.log

```

[lab3@sandy:/etc/rsyslog.d]$ ls
20-ufw.conf 50-default.conf  ids_alerts.conf
[lab3@sandy:/etc/rsyslog.d]$ cat ids_alerts.conf
# Log messages from the kernel module to /var/log/ids_alerts.log
:msg, contains, "[ALERT]" /var/log/ids_alerts.log

[lab3@sandy:/etc/rsyslog.d$]

```

```
Lab_5 — lab3@sandy: /var/log — ssh - vagrant ssh sandy — 143x68
Nov  5 20:49:59 sandy kernel: [ 1184.854189] [+]^C
lab3@sandy:/var/log$ ls
alternatives.log    auth.log.1    btmp.1      fontconfig.log   journal      speech-dispatcher tallylog    vboxadd-setup.log.3
alternatives.log.1  auth.log.2.gz  cups        gdm3          kern.log     syslog       unattended-upgrades vboxadd-setup.log.4
apport.log          auth.log.3.gz  dist-upgrade gpu-manager.log kern.log.1  syslog.1    vboxadd-install.log vsftpd.log
apport.log.1        boot.log     dpkg.log    hp             kern.log.2.gz syslog.2.gz  vboxadd-setup.log wtmp
apt                bootstrap.log dpkg.log.1  ids_alerts.log  kern.log.3.gz syslog.3.gz  vboxadd-setup.log.1 wtmp.1
auth.log            btmp         faillog    installer      lastlog     syslog.4.gz  vboxadd-setup.log.2
lab3@sandy:/var/log$ cat ids-bash: cannot create temp file for here-document: No space left on device
^C
lab3@sandy:/var/log$ cat ids_alerts.log
Nov  5 20:49:57 sandy kernel: [ 1183.607353] [ALERT] Malicious syscall: write to fd 7, content:
Nov  5 20:49:57 sandy kernel: [ 1183.607804] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.596540] He
llo world!
Nov  5 20:49:57 sandy kernel: [ 1183.607847] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.596540] He
llo world!
Nov  5 20:49:57 sandy kernel: [ 1183.607872] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.600947] Sy
scall_table addr
Nov  5 20:49:57 sandy kernel: [ 1183.607881] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.600947] Sy
scall_table addr
Nov  5 20:49:57 sandy kernel: [ 1183.607895] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.600948] si
zeof(unsigned lo
Nov  5 20:49:57 sandy kernel: [ 1183.607901] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.600948] si
zeof(unsigned lo
Nov  5 20:49:57 sandy kernel: [ 1183.607915] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.600948] si
zeof(sys_call_ta
Nov  5 20:49:57 sandy kernel: [ 1183.607921] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.600948] si
zeof(sys_call_ta
Nov  5 20:49:57 sandy kernel: [ 1183.608024] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.600981] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608031] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.600981] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608045] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.600991] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608051] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.600991] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608065] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601034] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608071] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601034] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608084] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601039] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608090] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601039] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608104] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601054] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608117] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601054] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608131] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601058] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608137] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601058] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608150] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601071] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608156] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601071] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.608169] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601086] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.608537] [ALERT] Malicious syscall: write to fd 2, content: lab3@sandy
Nov  5 20:49:57 sandy kernel: [ 1183.608583] [ALERT] Malicious syscall: write to fd 3, content: R??'t?w@D??'H??y??3 gq:S>?
Nov  5 20:49:57 sandy kernel: [ 1183.609257] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601086] [+]
l openat called
Nov  5 20:49:57 sandy kernel: [ 1183.609287] [ALERT] Malicious syscall: write to fd 7, content: Nov  5 20:49:57 sandy kernel: [ 1183.601118] [+]
l read called
Nov  5 20:49:57 sandy kernel: [ 1183.609295] [ALERT] Malicious syscall: write to fd 9, content: Nov  5 20:49:57 sandy kernel: [ 1183.601118] [+]
```