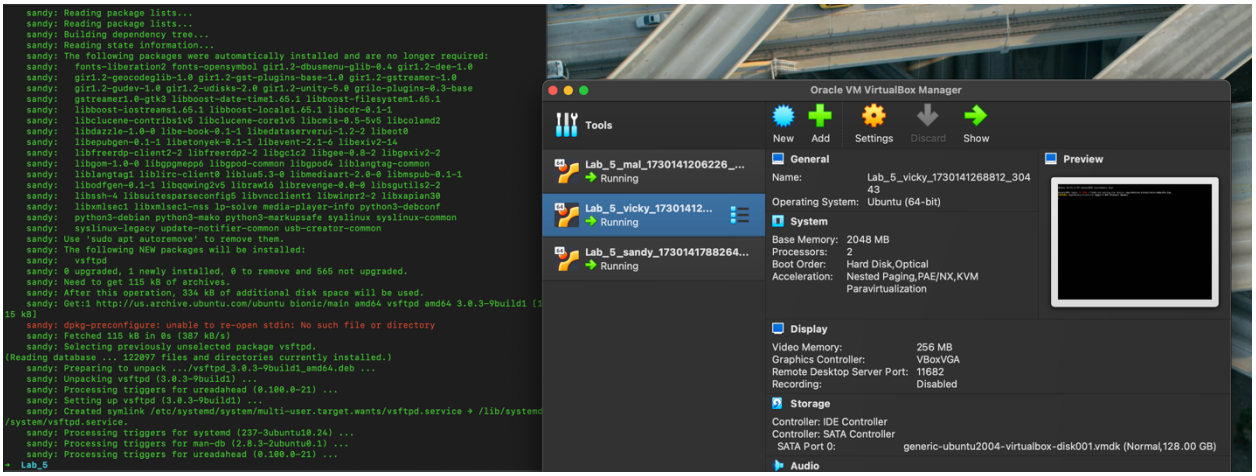
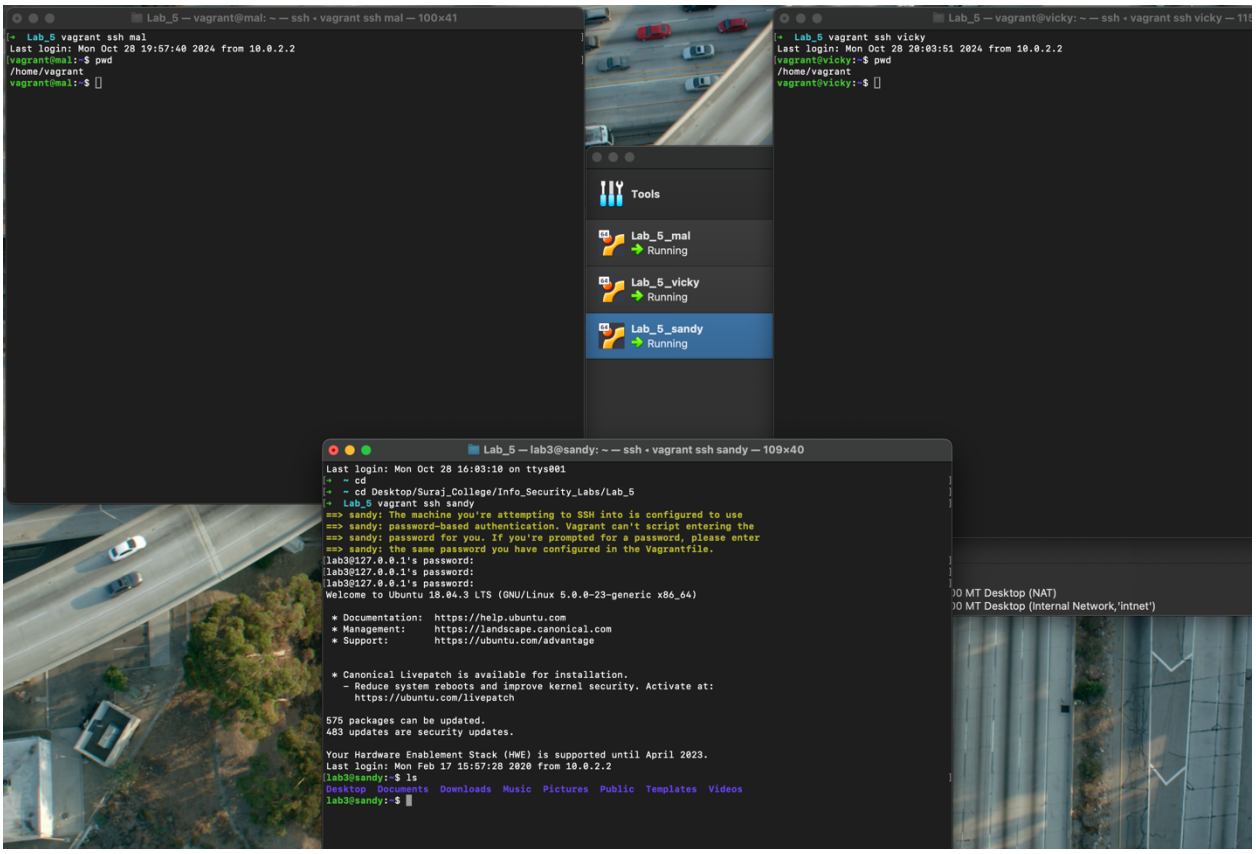
DIY Network-based IDS: Project 5

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

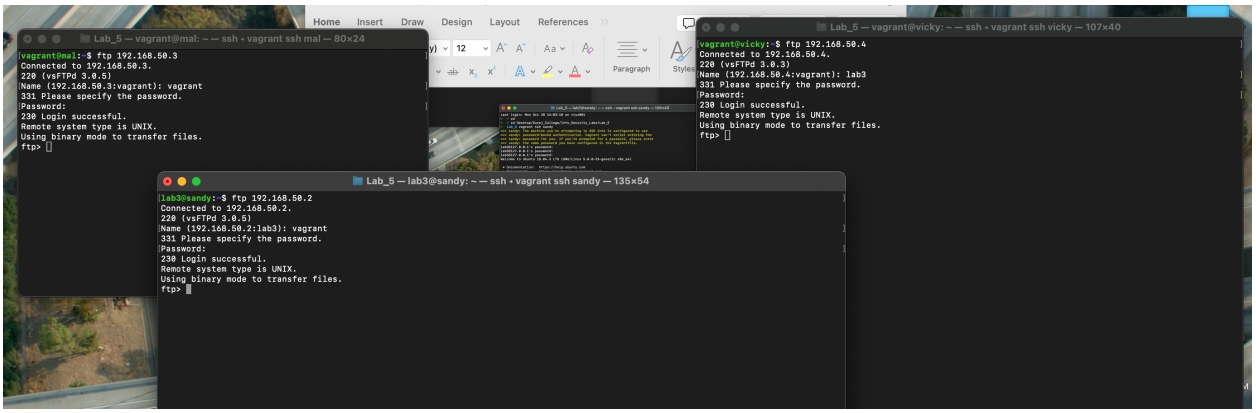


Afterward, I started each machine from my terminal using the command: vagrant ssh [hostname].

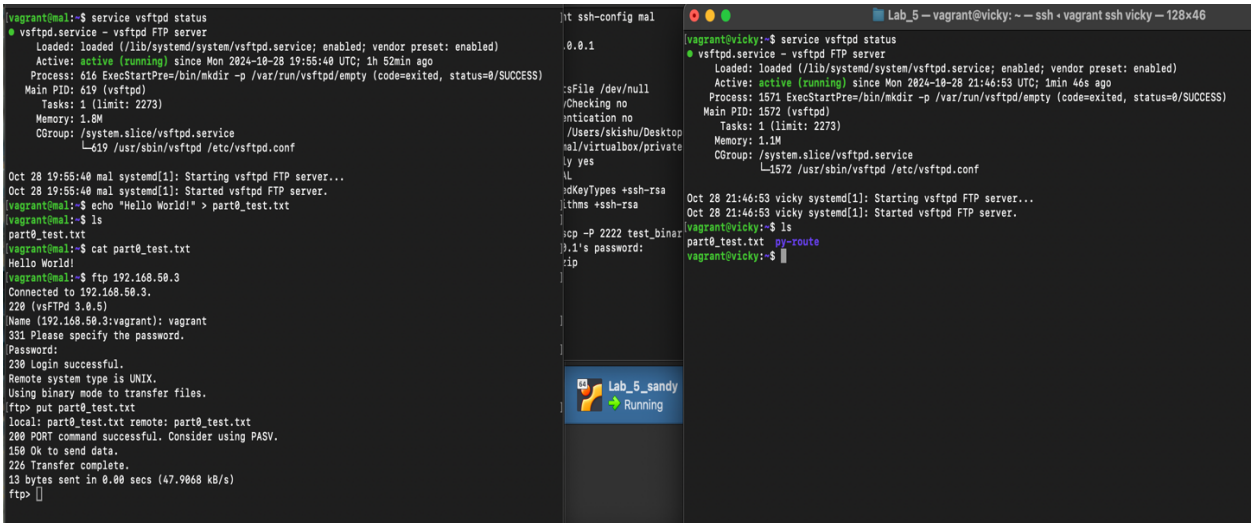


Part – 0:

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

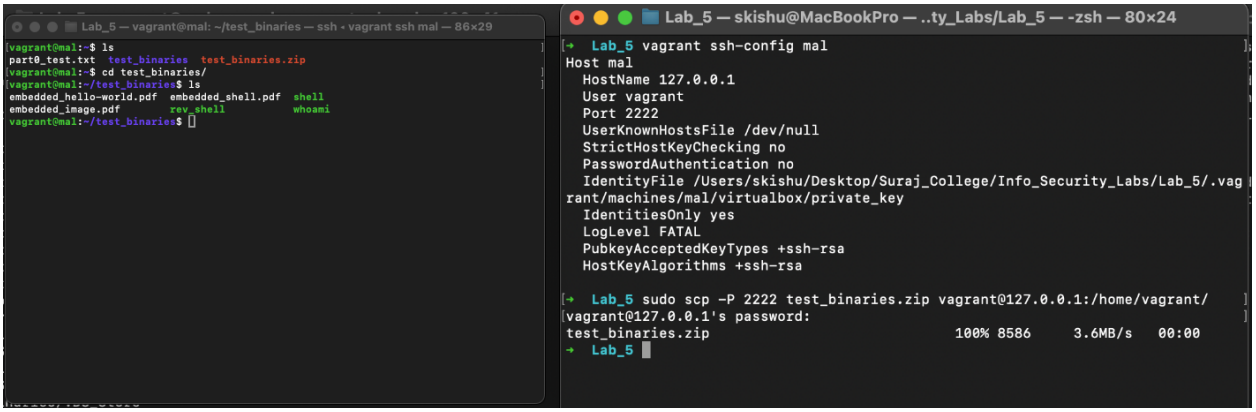


Then, I create a sample text file containing the text “Hello World!” in mal machine. After that I transferred this file to Vicky machine over FTP. First, I started the FTP process with “ftp 192.168.50.3” then entered the required username and password for a successfully login, and sent the file named 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.

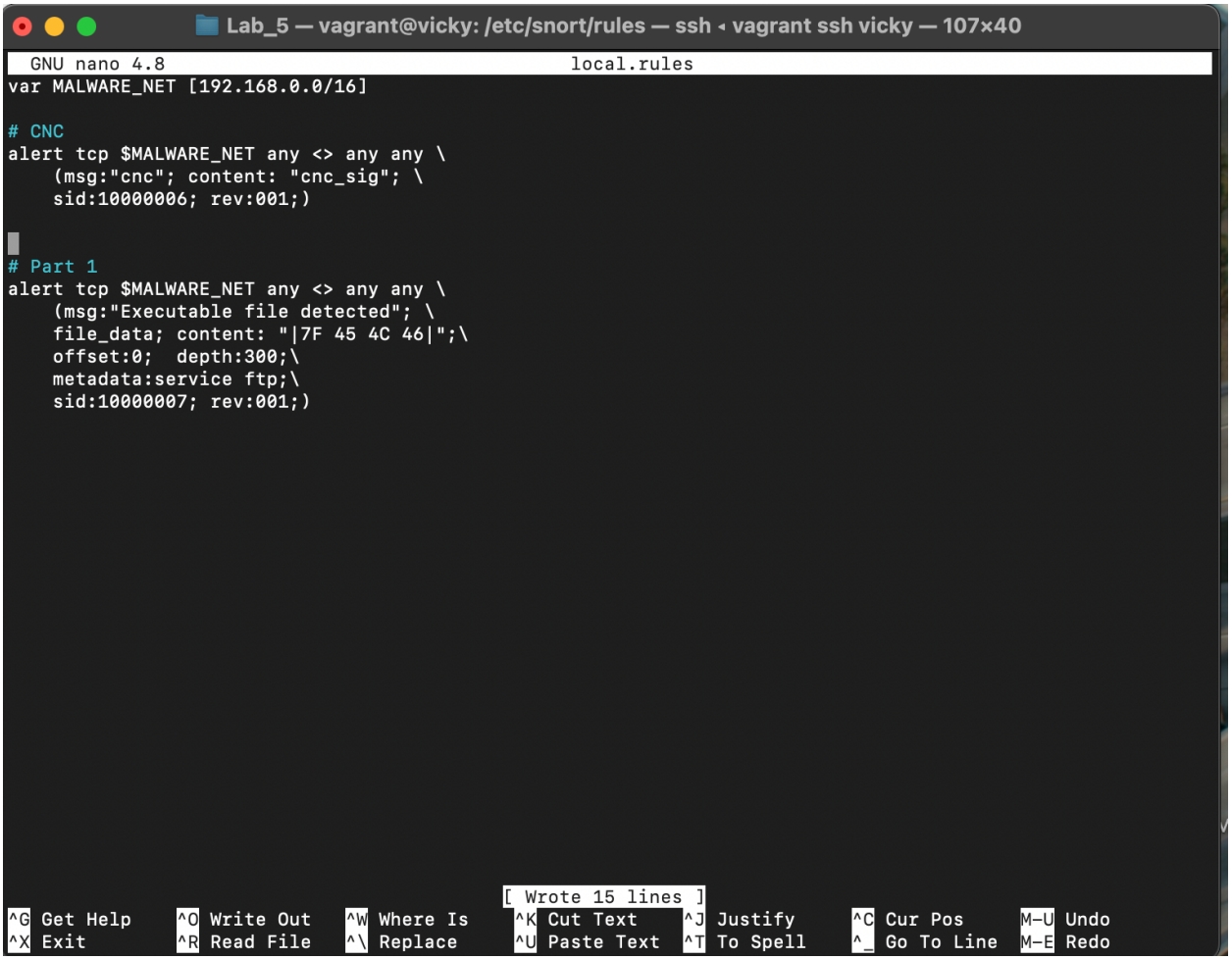


Part – 1:

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.

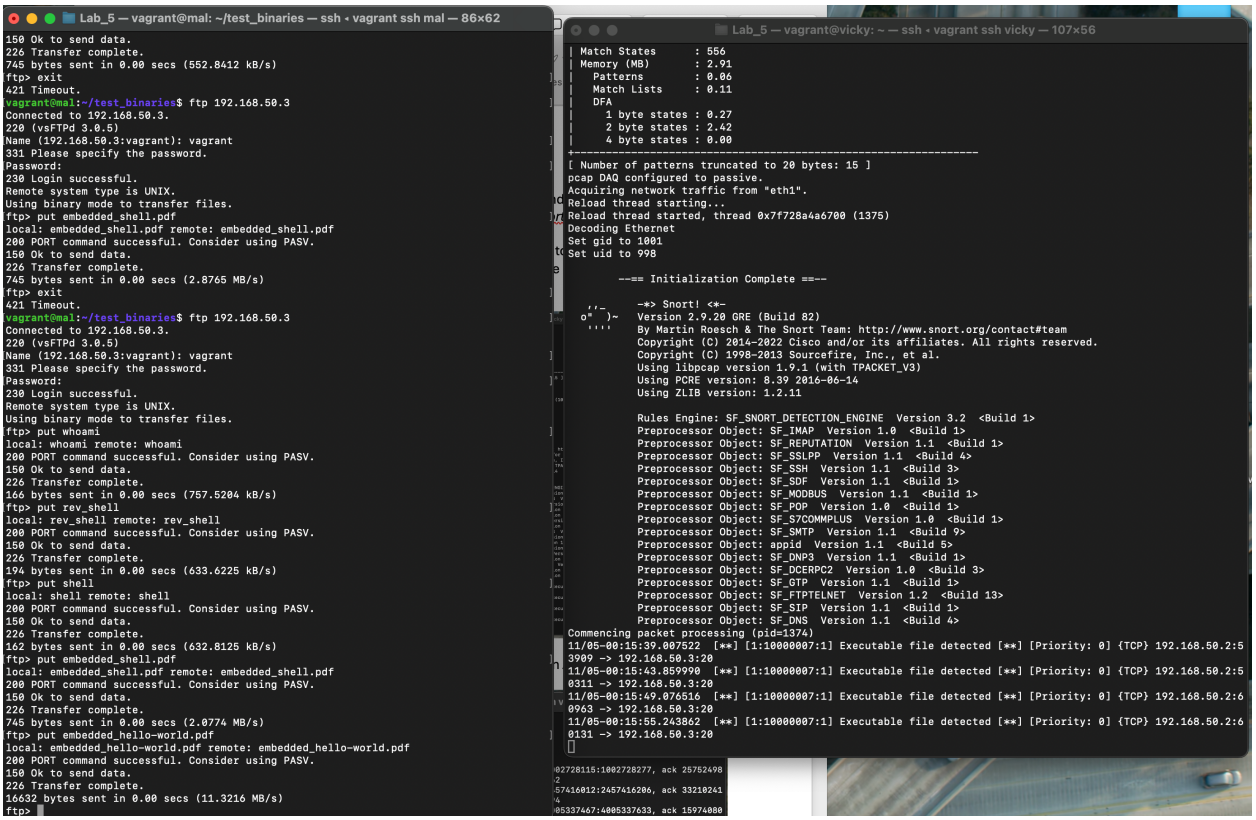


On the Vicky machine, I added a new Snort rule in the /etc/snort/rules directory under the local.rules file to detect executable files transmitted over the network. I also enabled write access on Vicky to avoid permission issues.

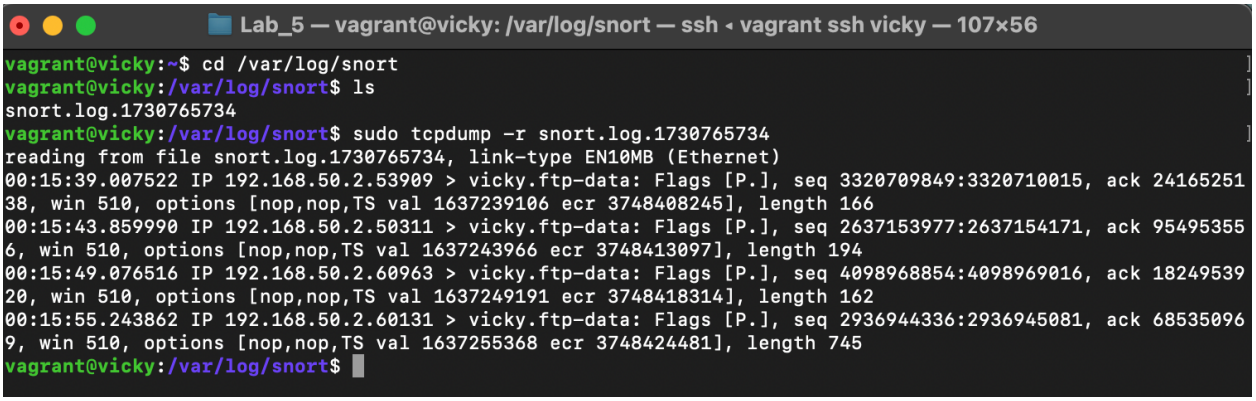


Then, I established a Snort console on Vicky using this command:
sudo snort -A console -i eth1 -u snort -g snort -c /etc/snort/snort.conf

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.

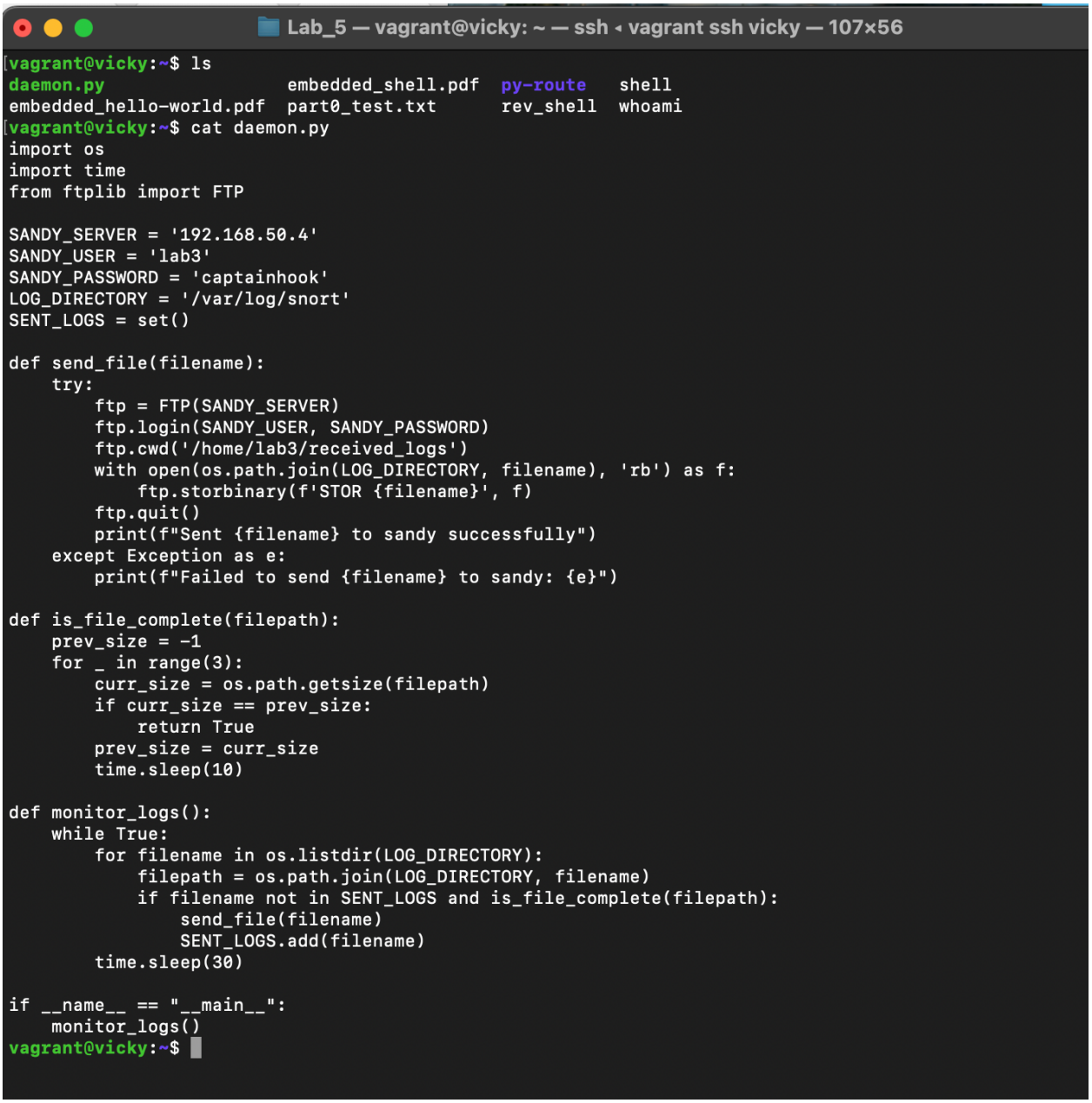


Snort saved all the log of file transfer and save in the file in /var/log/snort directory.

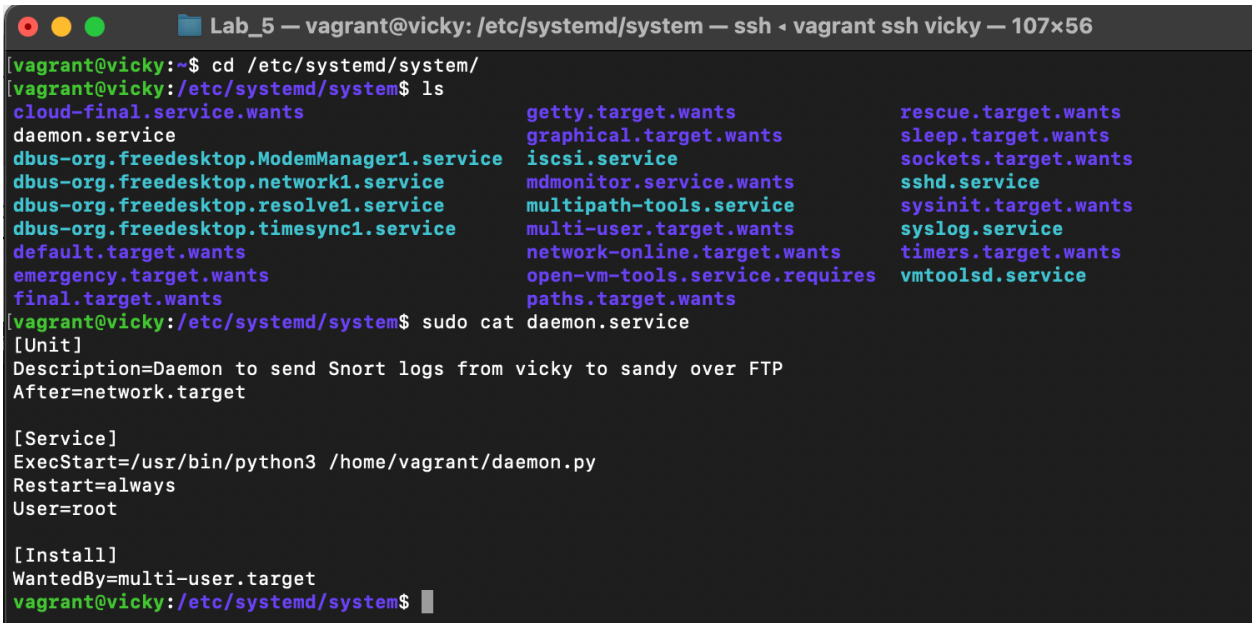


Part – 2:

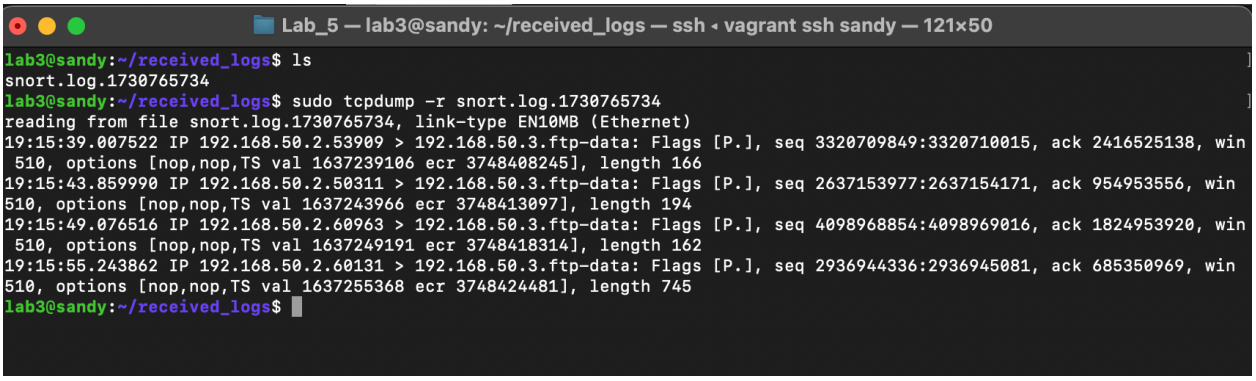
In this part, I created a Python daemon script named daemon.py to automate the process of sending Snort log files from the /var/log/snort directory on Vicky to 'Sandy' whenever a new log file is generated.



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



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.



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 ◂ vagrant ssh sandy — 143×68

```
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");
```

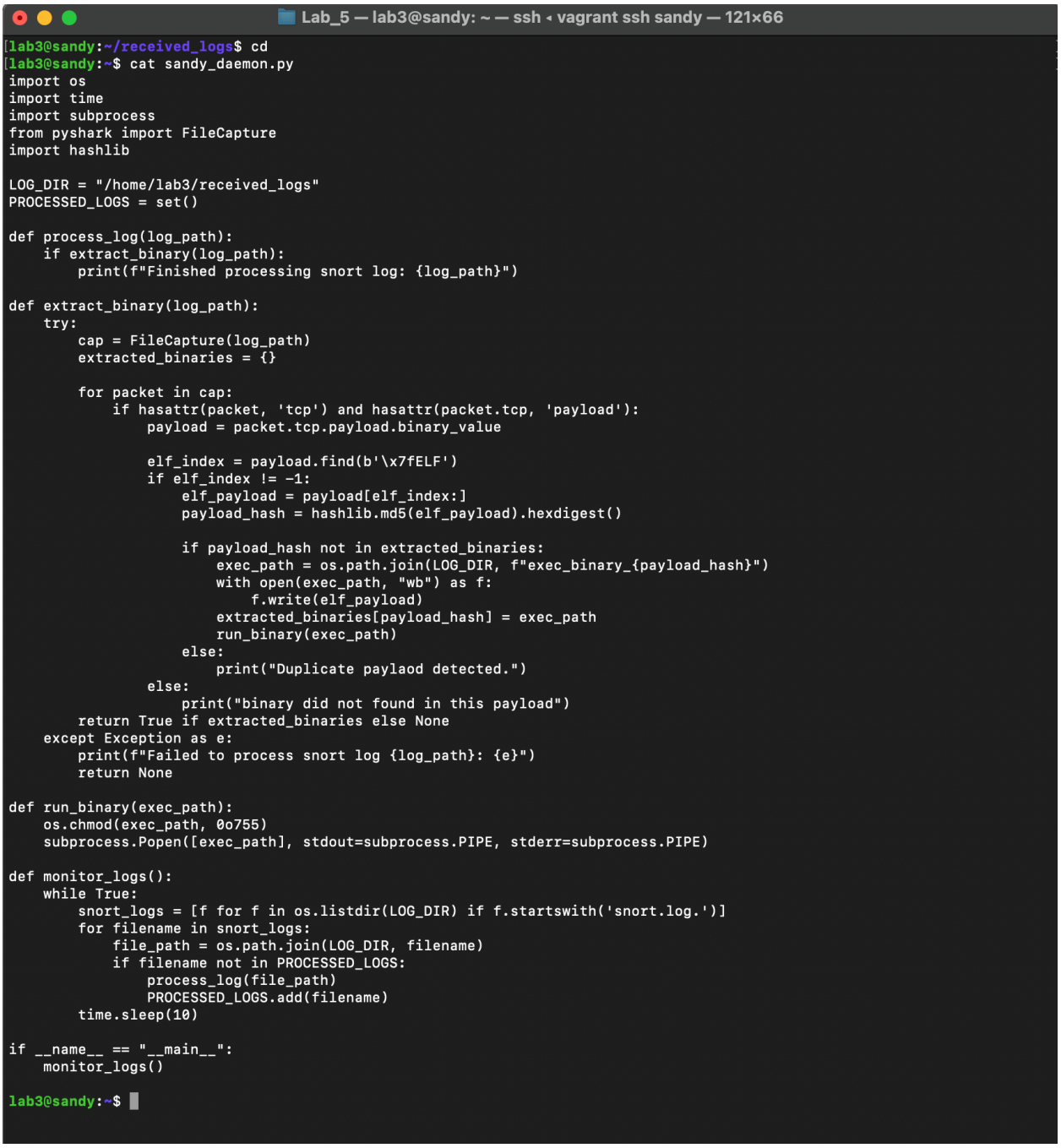
I compiled the code by running make all in the src folder, which created the required files, and then I inserted the module using:

Lab_5 — lab3@sandy: ~/Desktop/lab_files/src — ssh ◂ vagrant ssh sandy — 176×65

```
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.



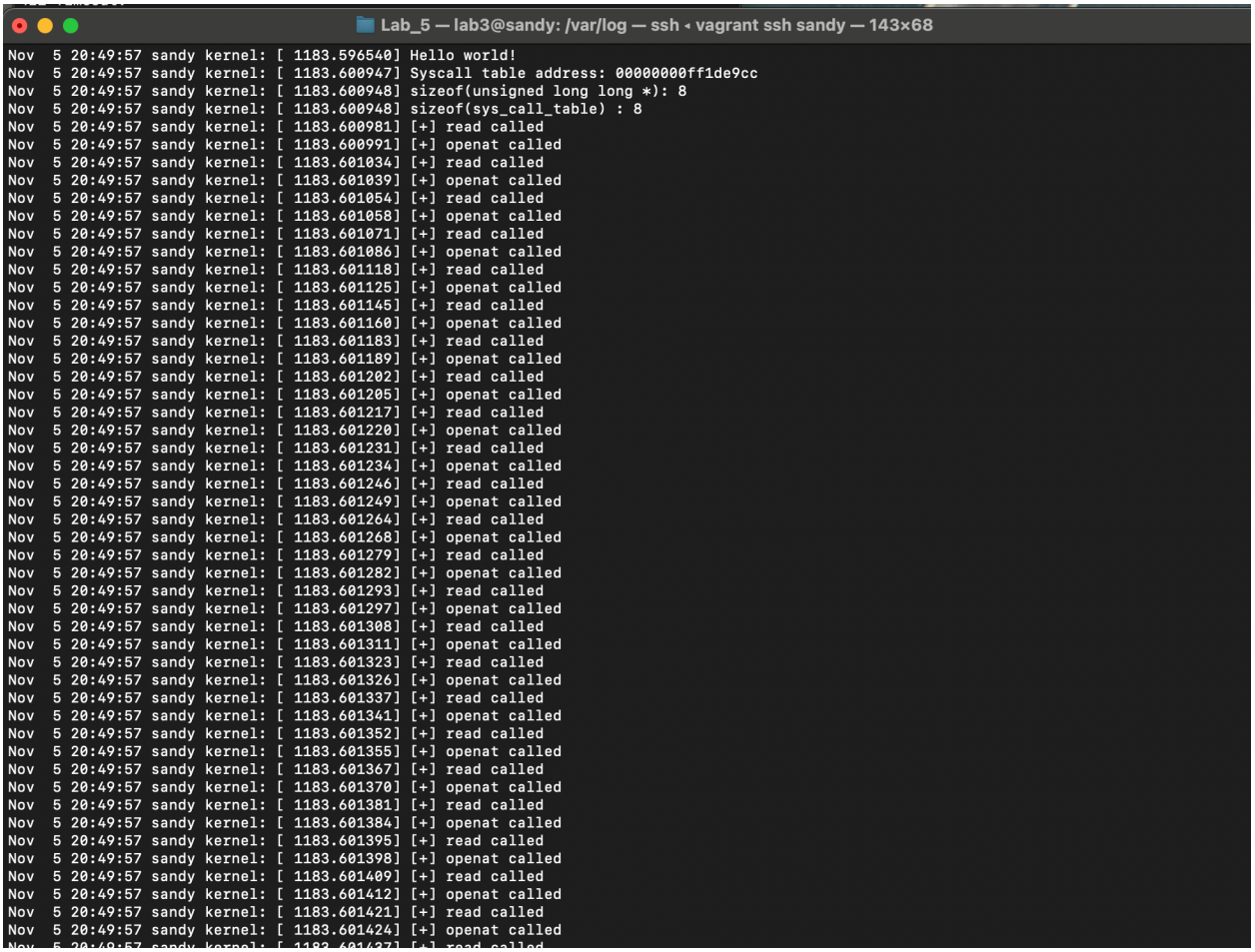
Also, I created a systemd service, sandy_daemon.service, in /etc/systemd/system to run sandy_daemon.py in the background continuously.

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

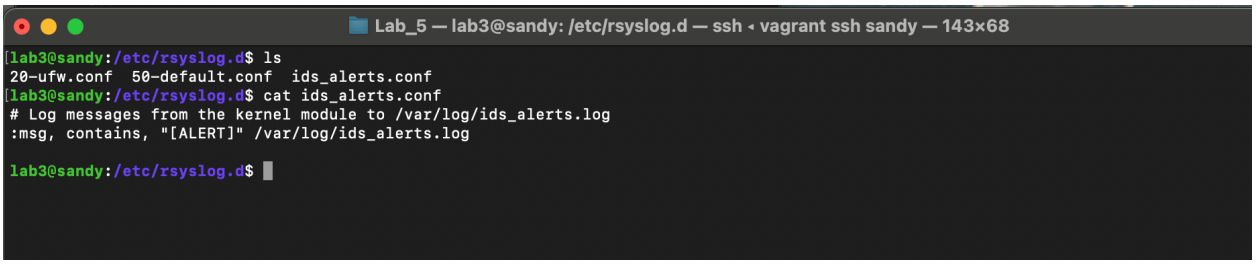
```
Lab_5 — lab3@sandy: ~/received_logs — ssh ‹ vagrant ssh sandy — 121×66
[lab3@sandy:~$ ls                                                                    ]
 binary_files_from_daemon  Documents  Music           Pictures  received_logs    Templates
 Desktop                   Downloads  part0_test.txt  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$
```

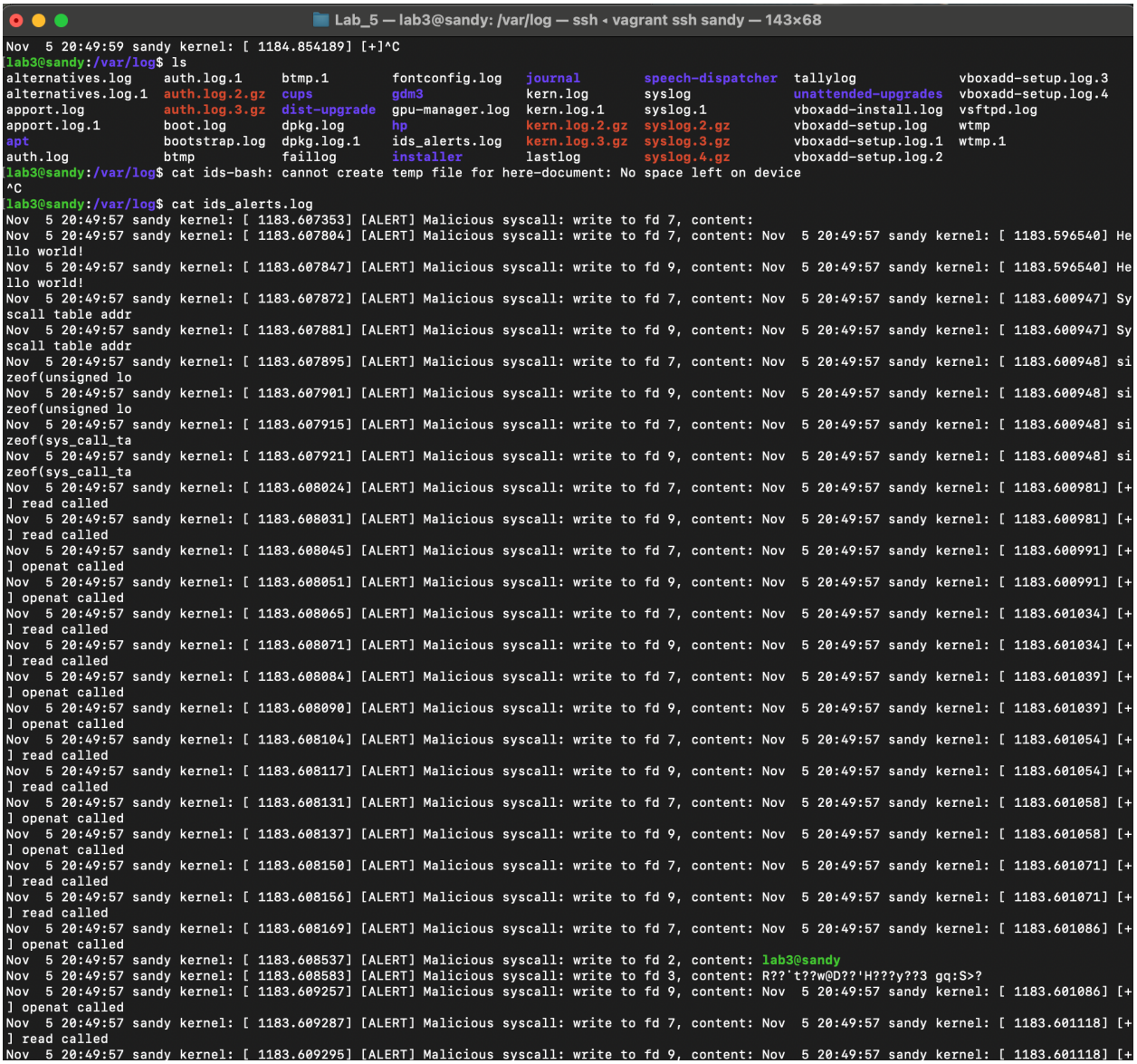
After running binaries through the daemon, malicious syscalls log details are capture in /var/log.kern.log file.



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




Lab_5 — lab3@sandy: /var/log — ssh ‹ vagrant ssh sandy — 143×68
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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+
] 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] [+