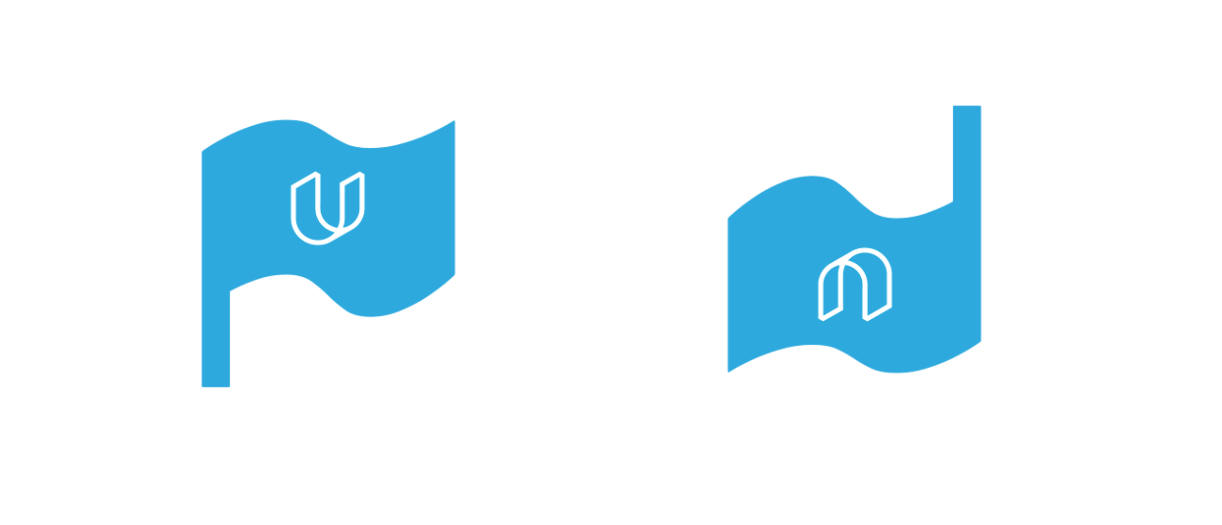
### **Responding to a Nation-State Cyber Attack**



**By: Moustafa Darwish**

**Scenario**

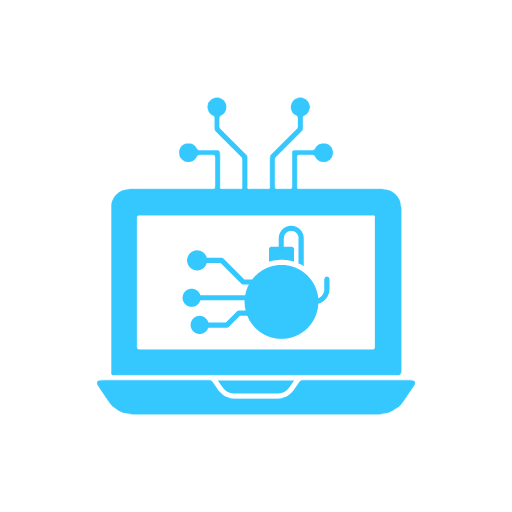
**South Udan** is a small island nation that is a peaceful and technologically advanced nation! Because of its small size, the country believes in efficiently using its land and natural resources. Their scientists recently came up with a novel and cleaner means of performing nuclear fission of an element 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.

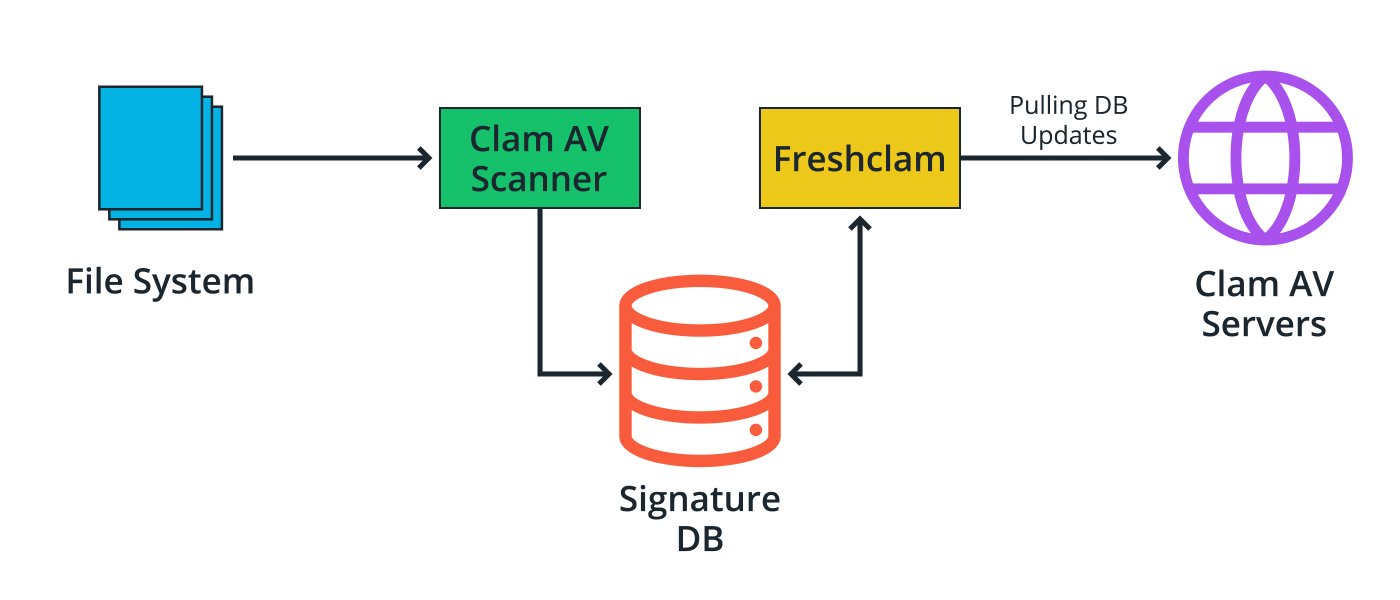
**Section 1**

**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.

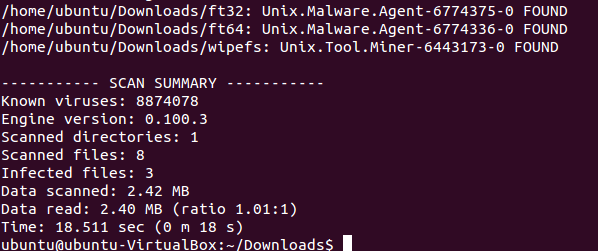


**The below command is used to scan the infected directory:**

**clamscan -i -r /home/ubuntu/Downloads**

-i tells ClamAV to only display infected files. The -r flag makes the scan recursive to ensure that it scans files in all associated directories and subdirectories.

**Below is the result of the scan:**

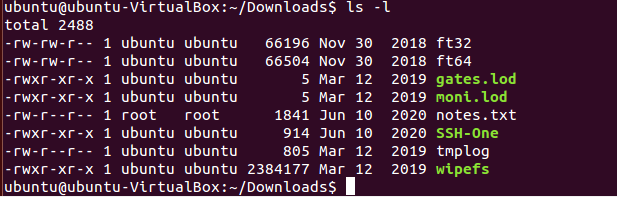


As we can see 8 files have been scanned and 3 of them have been recognized by the ClamAV as a malicious file.

Now our next step is analyzing the undetected files manually to make sure they are safe and not a zero-day attack or malware file or even a specified attack with a new signature that wasn’t identified by our malware scan.

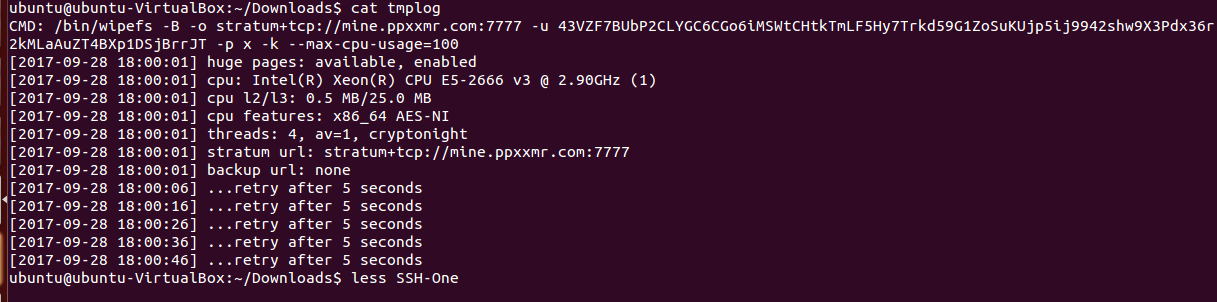
### **Task 2: Suspicious File Identification**

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

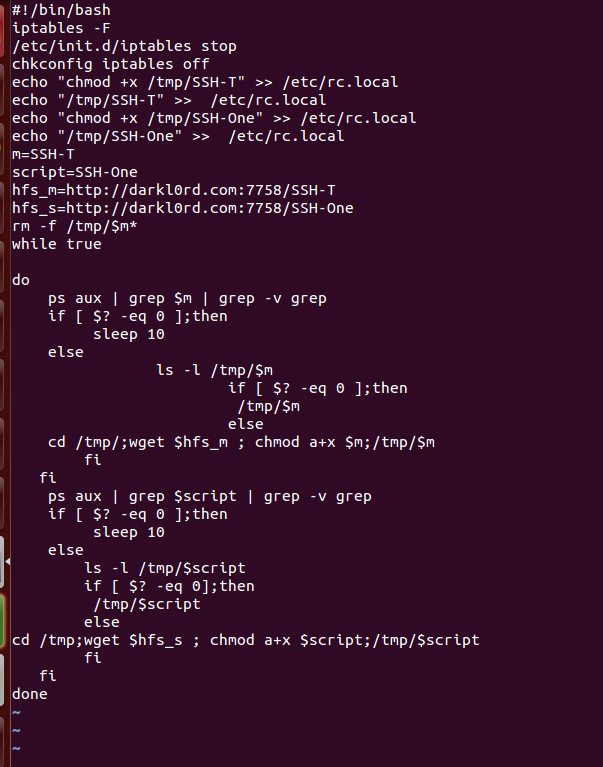


The file **notes.txt, moni.lod, gates.lod** did not show anything of interest. Additionally, when analyzing the **tmplog** file I found something interesting.

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:



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.

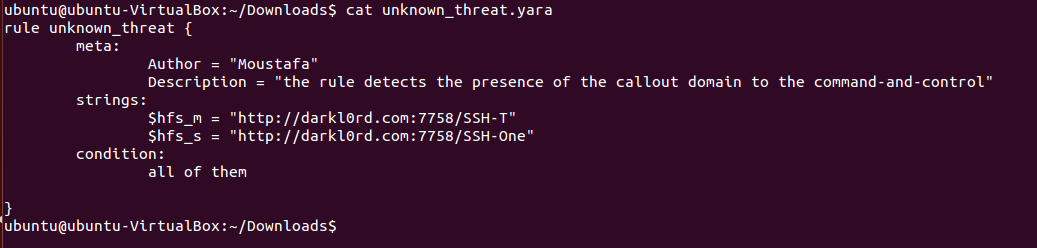
We can later see he made 4 variables **$m, $script, $hfs\_m, $hfs\_s**. The contain the name of the scripts and the command-and-control callout domain. And after them a command that removes the script from the **/tmp** directory as an attempt to hide evidence.

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

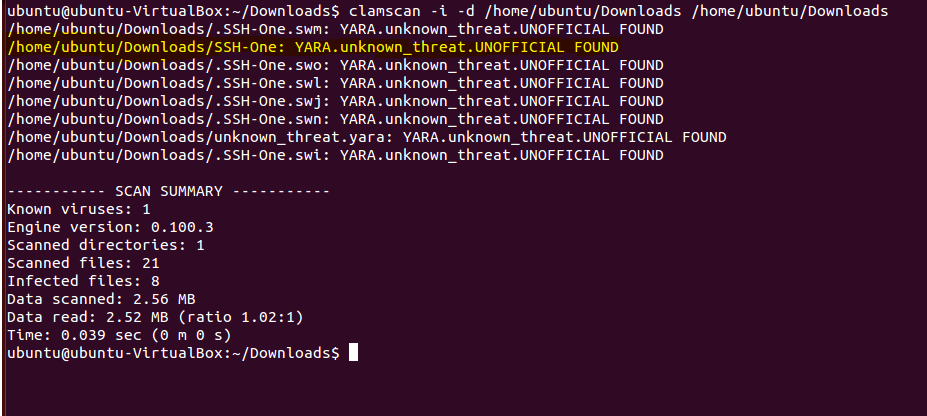
Now, our next task is to make our own custom **YARA** rule that contains the string pattern we have found in the malware file and create a signature for the **ClamAV**, so that when we scan our other servers and the same file exists, we will be able to identify it.

### **Task 3: Yara Rule Creation**

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



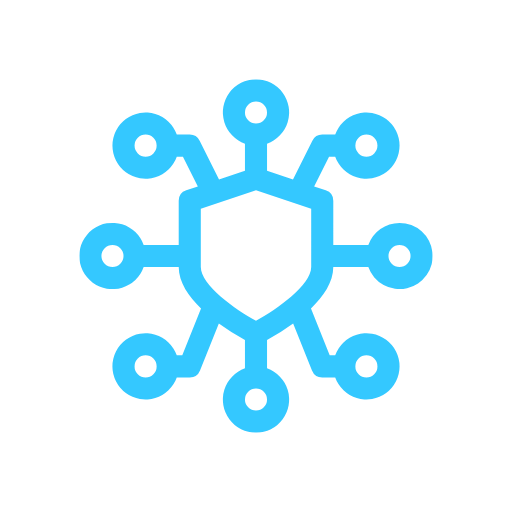
For the matching string pattern, the callout domain to the command-and-control has been added. For the condition, all of them must be present for the identification to trigger.



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**

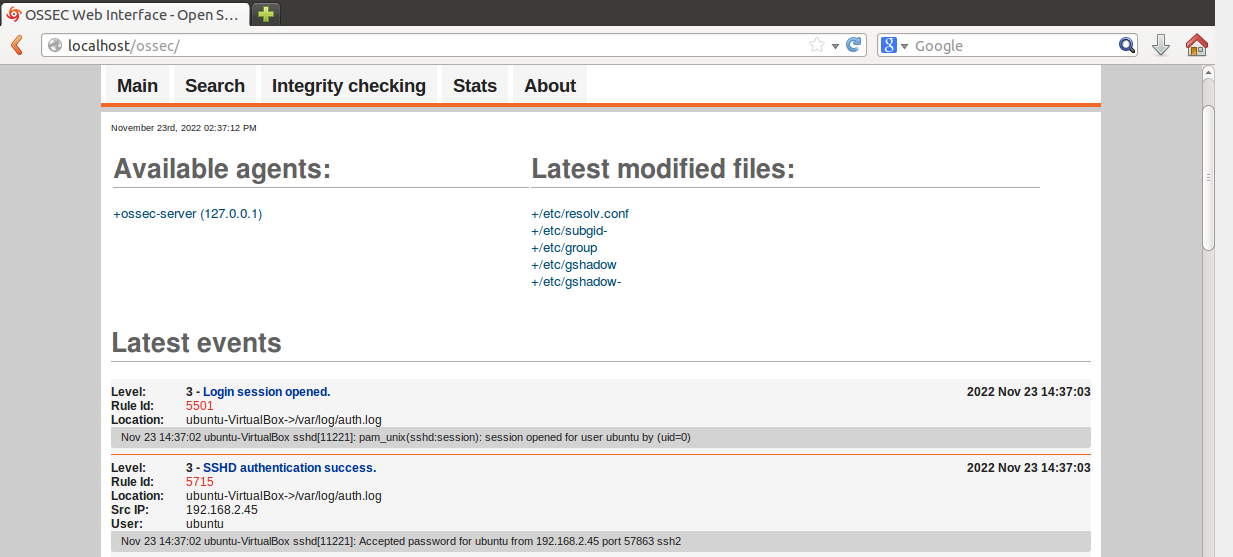
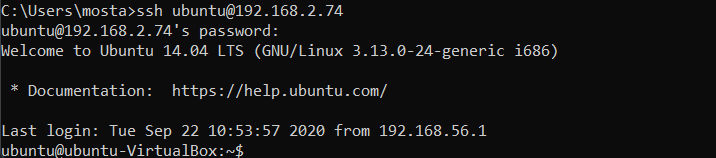


### **Task 1: Implement HIDS**

### **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.

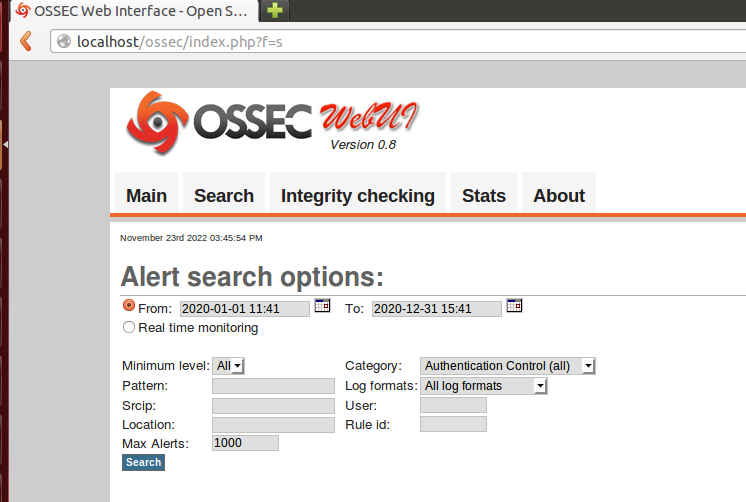


We can see that when we attempted to connect to the system via SSH, the HIDS successfully logged the event proving that it works in real-time.

**Task 2: Locate Suspicious IP**

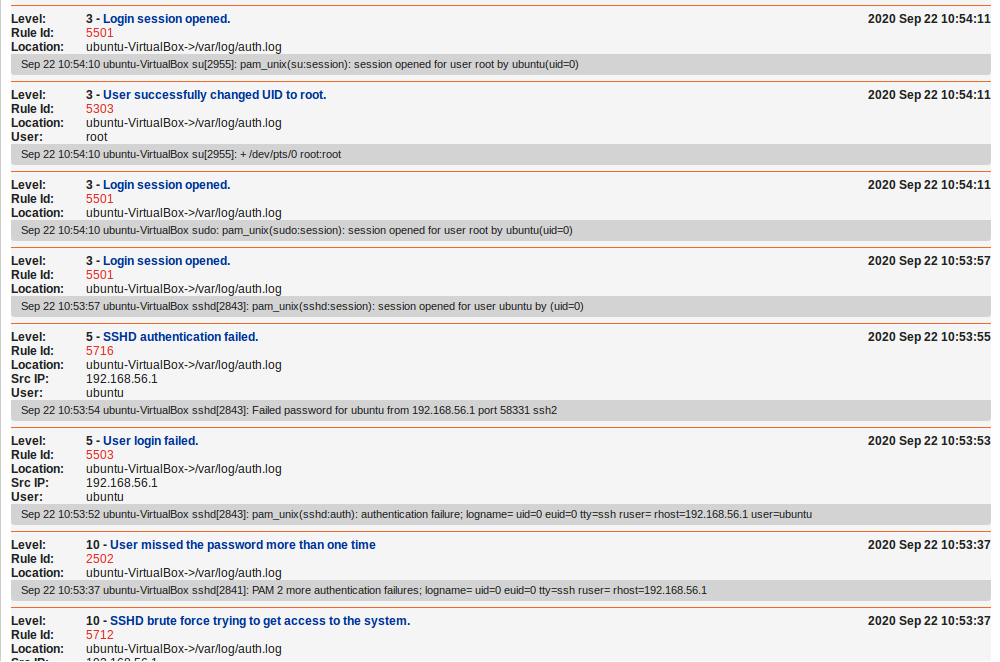
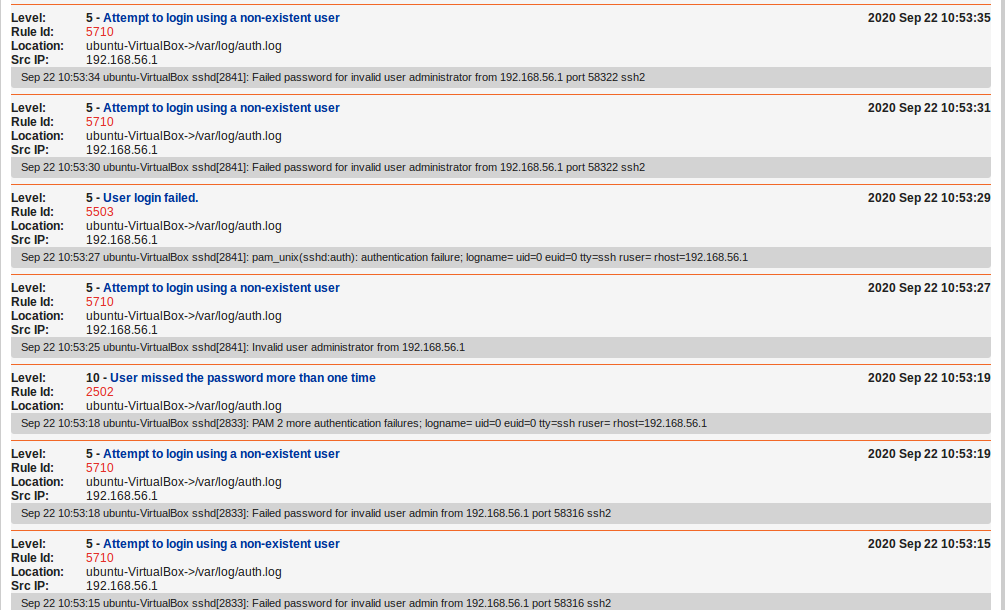
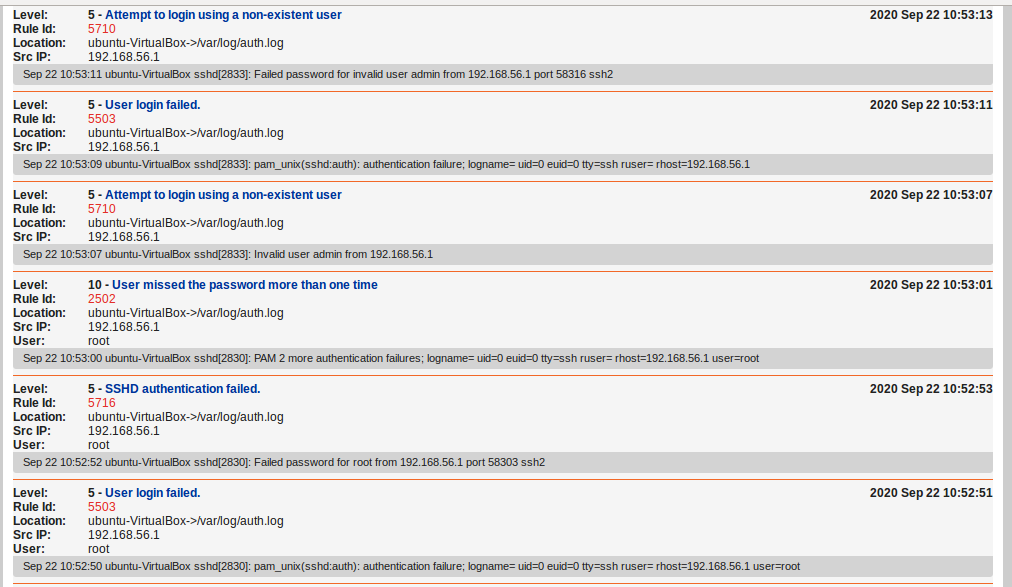
Now we want to collect the next **indicator of compromise (IoC)**, which is the IP address that attempted to break into the system. This **IoC** will prove significant in attributing the threat actors behind the cyber-attack. We were told that the initial security alarm consisted of multiple failed login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



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:



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**

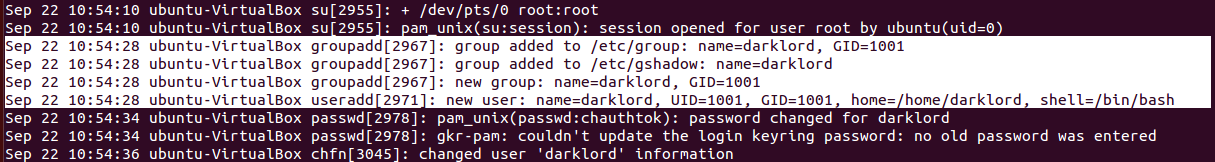
Once the attackers managed to break into the system, they created a backdoor username and launched a process that allows them to log in through a non-standard port number.

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

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.

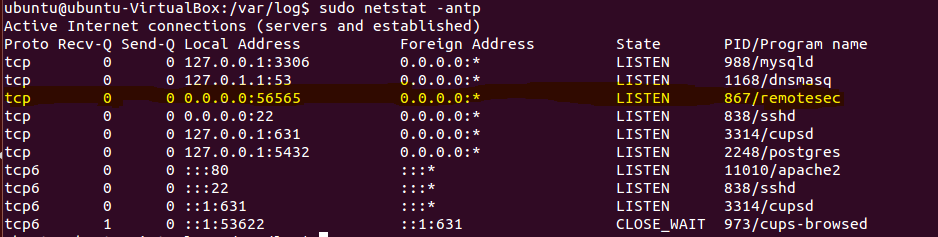


By analyzing the date, time, and location, we can use this information to further investigate the **auth.log** file to find the backdoor username.



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**.

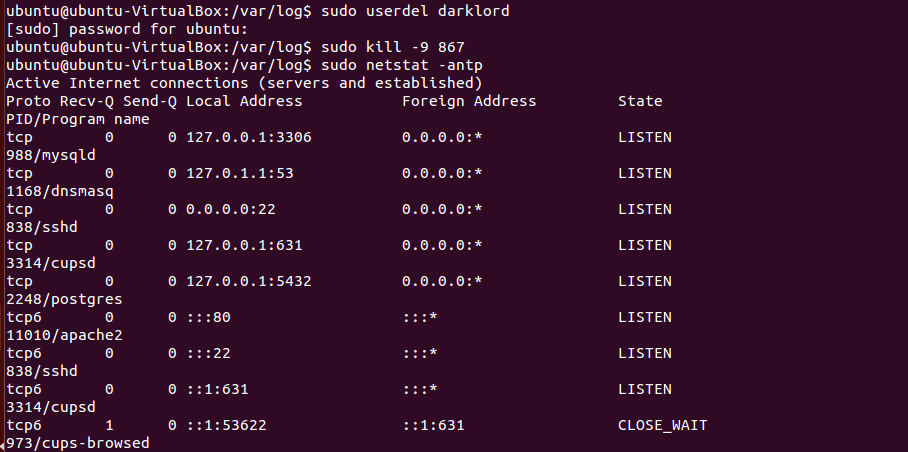
Now, we have to locate the malicious process that allows the attacker to log in. To do so we used the **netstat** tool to view all the inbound and outbound connections of the system:



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.



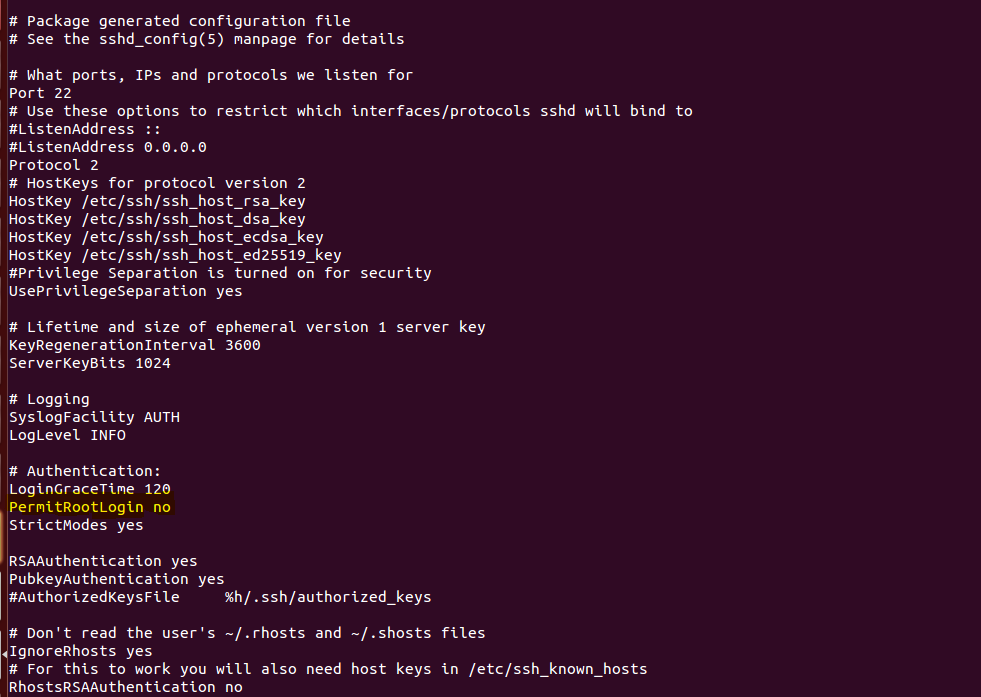
The port number used by the attacker as a backdoor to login to the system is **56565.** Further investigating this port on the internet, it appears to be a known port that is used by the attackers for backdoors attack. Reference: <https://www.speedguide.net/port.php?port=56565>Now, we delete the rouge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

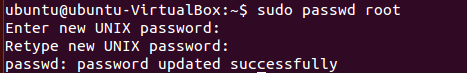
We should also restrict root level login for remote users. This allows us to narrow down the blast radius in case of a credential compromise and prevents complete system takeover.

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



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**



**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**

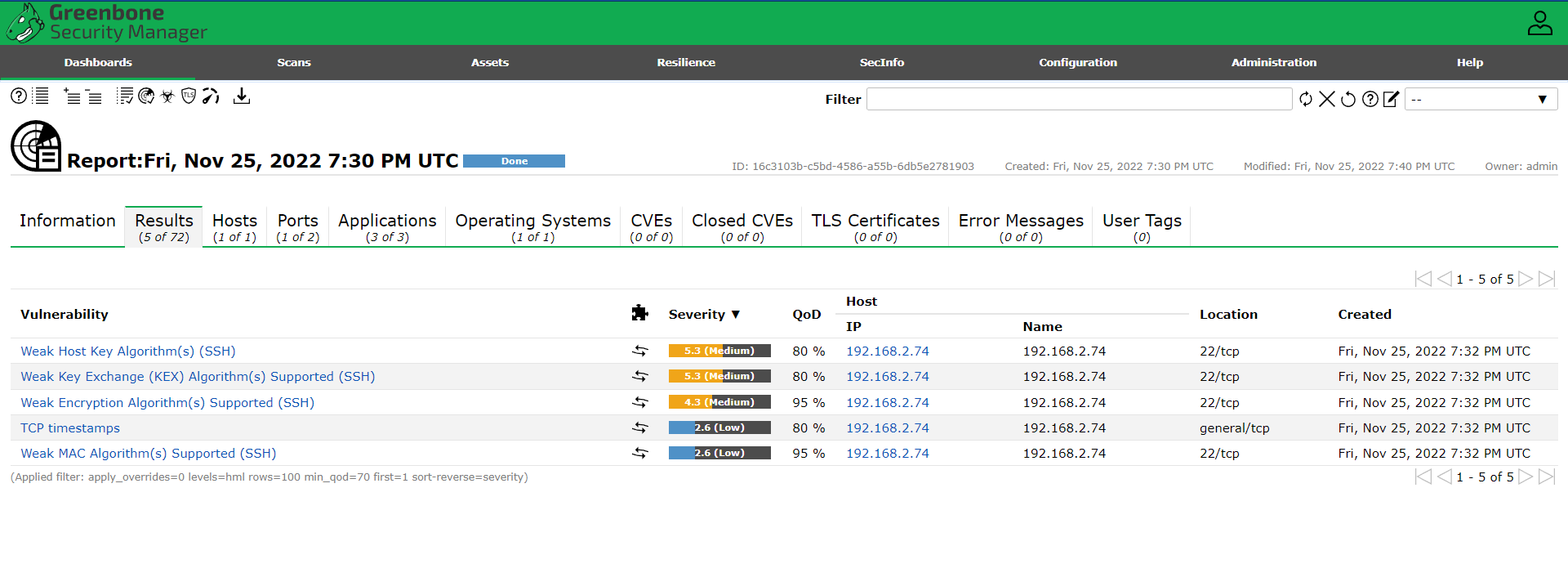
**System Hardening for Enhanced Security**



### **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.

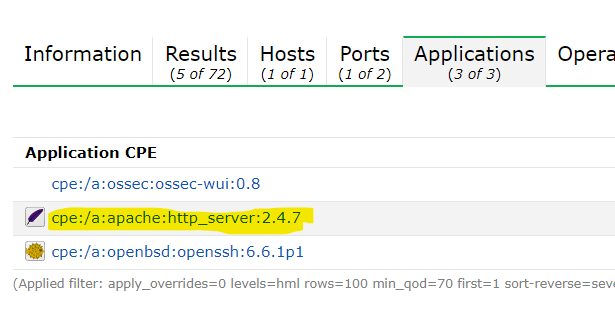
Below is the report generated after scanning the system:



### 

### **Task 2: Patching Apache**

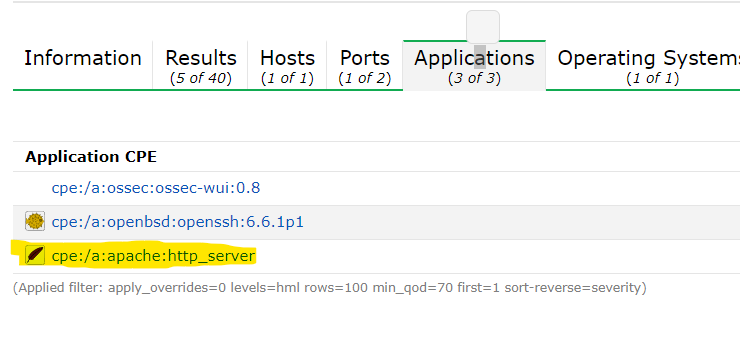
After being provided with the report on the existing vulnerabilities on the system, we can notice that the jump host is also running an Apache HTTP server which can be accessed from the internet and can serve as an attack point in future incidents. To harden the Apache server, we must remove the version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



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**



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.

**Configurating 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