Nessus Vulnerability Management Lab
By Michael Ambeguia

Purpose:

To provide myself with a beginner friendly, structured, and hands-on learning experience applying the vulnerability management process using Nessus Essentials. I would also like to practice effectively identifying, analyzing, and remediating security vulnerabilities within Linux-based environments.

Skills Applied:

1. Applying the vulnerability management process
2. Using the Nessus Essentials vulnerability scanner
3. Configuring/ Maintaining vulnerability Scanners
4. Interpreting scan results
5. Remediating vulnerabilities
6. Reporting Findings

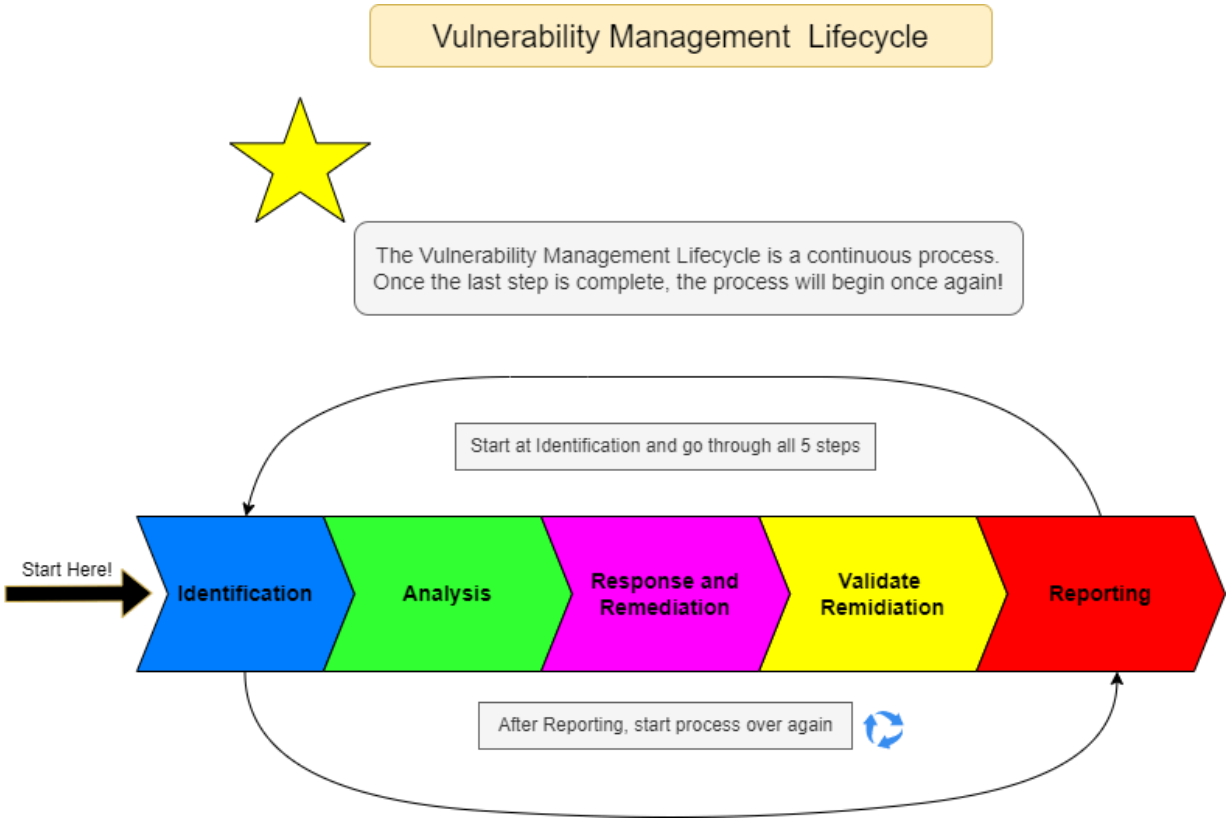
Sections:

Section #1 Lab Introduction:

1.1. Lab Topology:

All devices for this lab are on the same subnet, hence the circle containing the topology of the lab. The devices are on the 192.168.1.0/24 subnet.

1.2 Vulnerability Management Lifecycle:



The vulnerability management lifecycle has 5 steps.

Step 1. Identification: Identification is when you use vulnerability scanners to scan devices and applications for vulnerabilities. At this step your goal is to use the tools at your disposal to find ***possible*** vulnerabilities. Not all vulnerabilities are valid. Some might be false positives which means that the scanner thinks it found one when it is not present. Some might be false negatives which means that the scanner thinks a vulnerability is not present when it is. That is why step #3 is so important.

Step 2. Analysis: Analysis is the step after the scanning occurs. During this step your goal is to analyze the results of the scan. At this step you will analyze the vulnerabilities found and prioritize which ones need to be remediated. This is done based off of the CVSS scores and the ease of the remediation.

Step 3. Response and Remediation: During this step you will use a remediation strategy to resolve the vulnerabilities. The remediation strategy I will be using is to first verify that the vulnerability is present, patch the vulnerability, then rescan the device to verify that the vulnerability has been remediated. If the vulnerability is still present you will have to try patching it again.

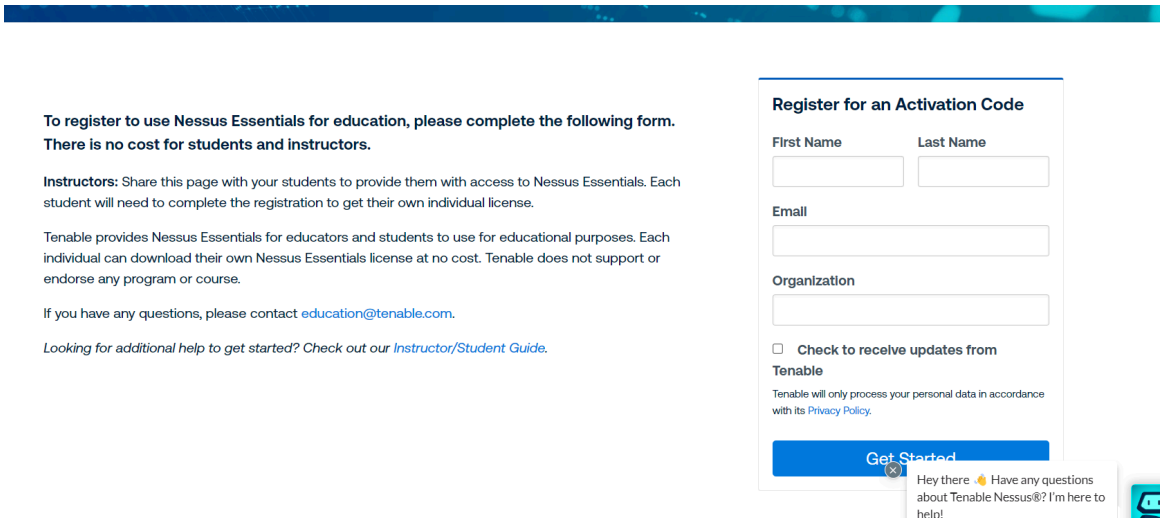
Step #4: Validate Remediation: This step is when the rescanning to validate the vulnerability is gone. In this lab I included it as part of step #3 since it is closely tied to step 3.

Step #5: Reporting: The last step is to report your findings. This step also serves as a point of reflection. You can go over the steps you took during the scan and think about what went wrong and what you might do next time to improve the process.

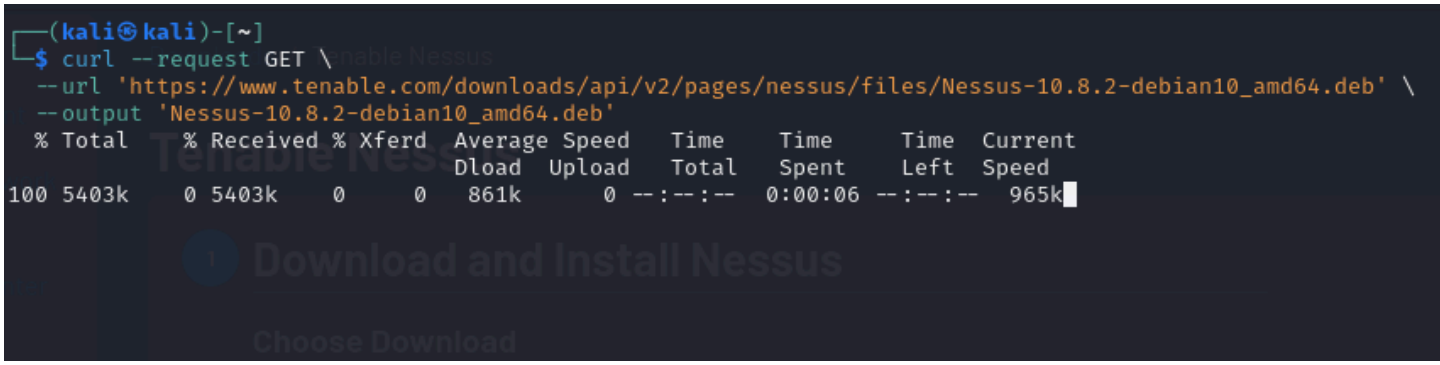
Section #2 Nessus Installation:

In this lab I will be using Nessus Essentials, the free version of Nessus Professional, to perform the vulnerability scanning.

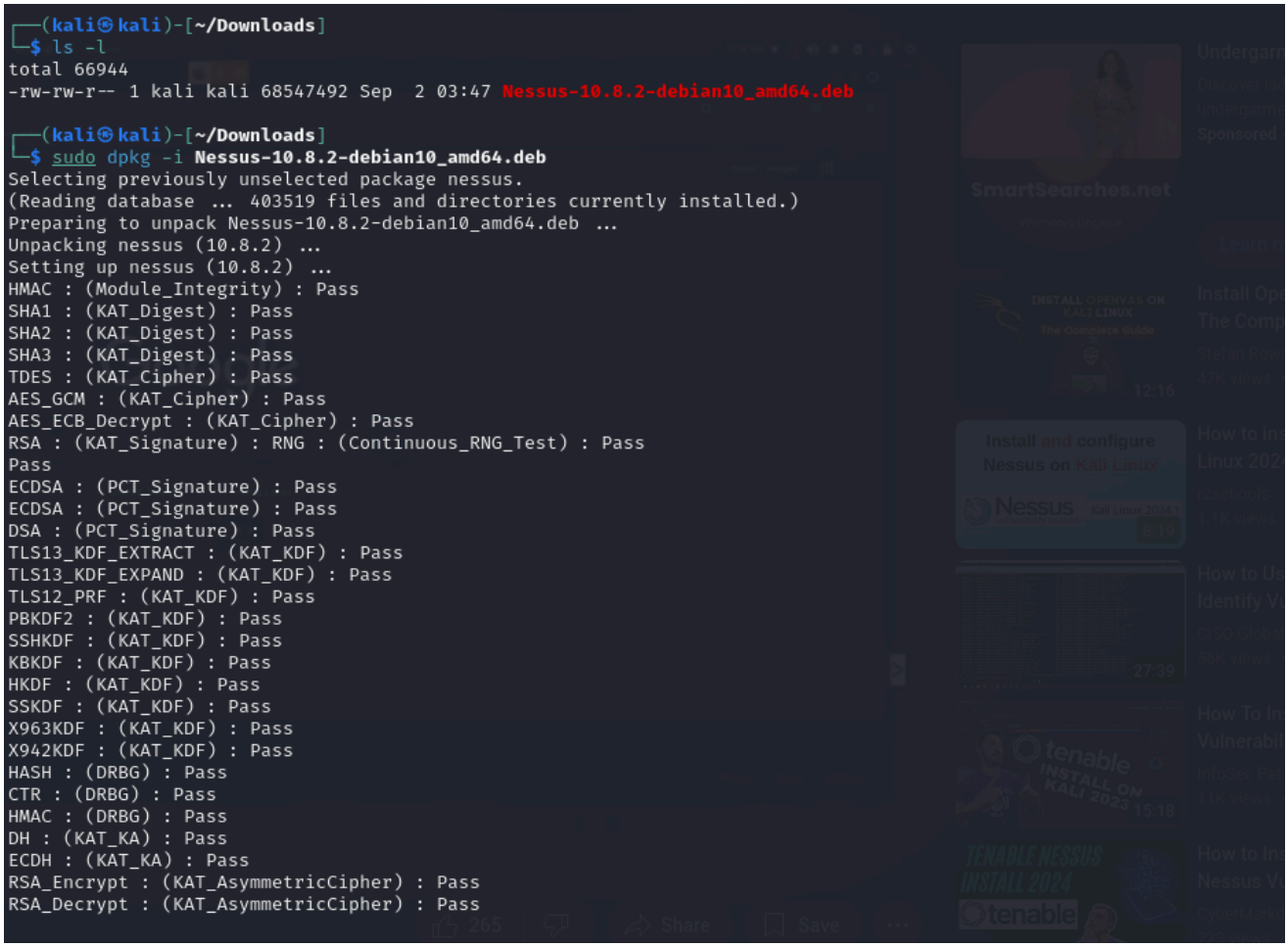
2.1 Sign up for a Nessus For Education account on the Nessus website: https://www.tenable.com/tenable-for-education/nessus-essentials?edu=true so that you can get a license for Nessus Essentials.



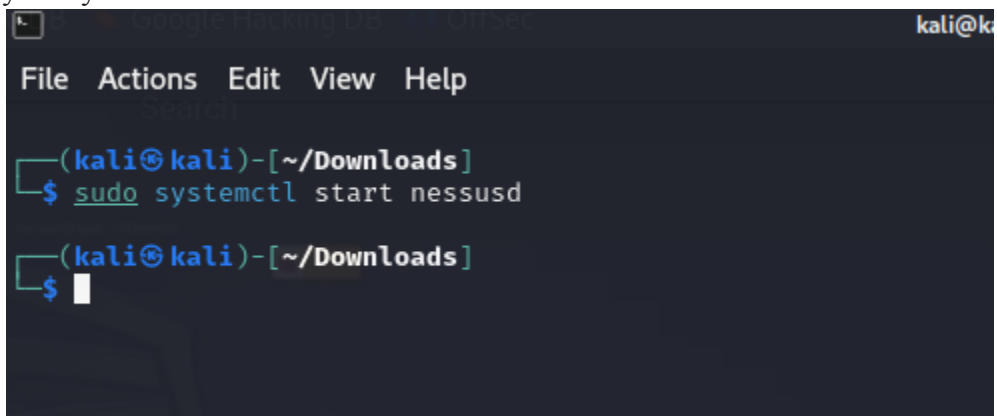
2.2 Download Nessus



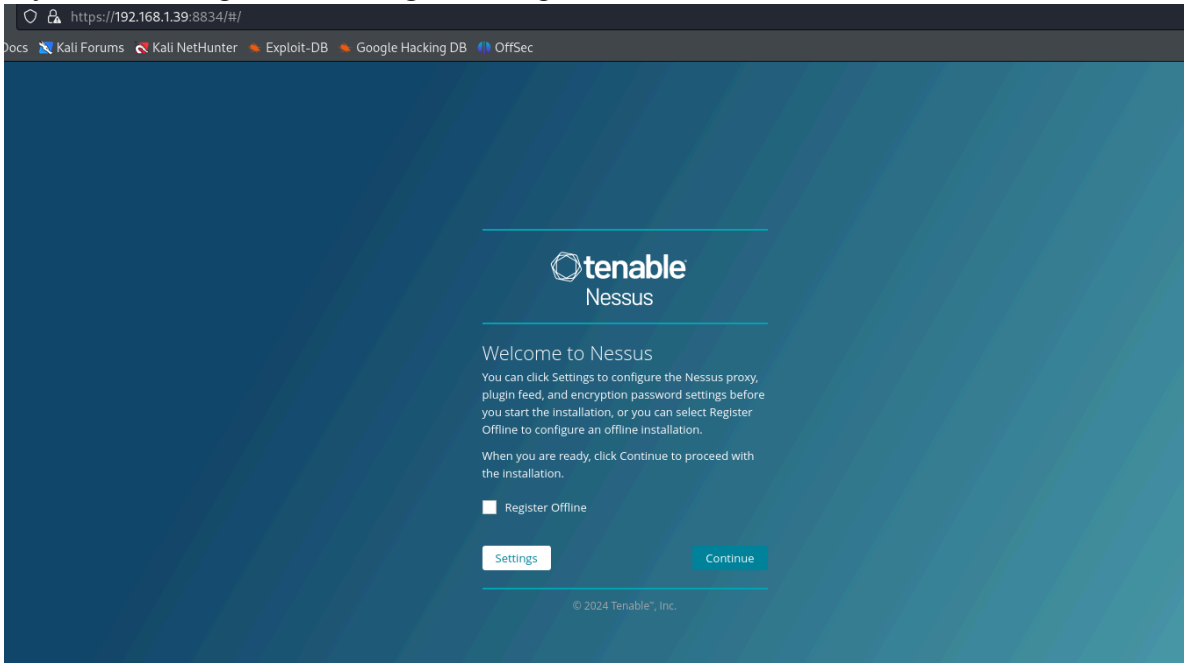
I used dpkg -i to install nessus. Dpkg -i is different from apt install since apt install is used to install packages from a remote repository over the internet while dpkg -i is used to download packages you download manually off of the internet onto your Debian based Linux device.



Once it is installed,, you need to start the nessusd service so that Nessus is up and running on your system.

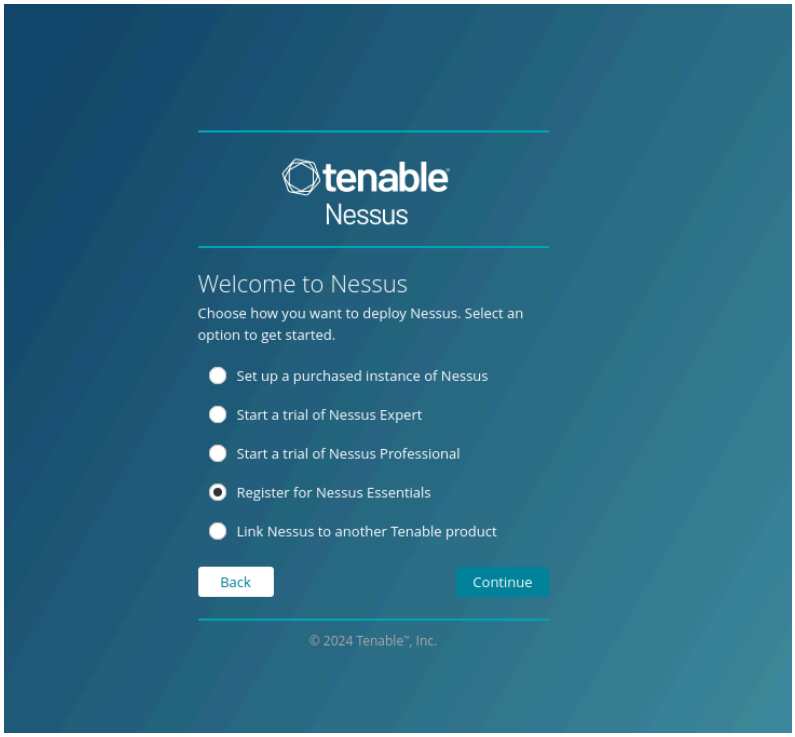


Nessus can then be accessed through your browser on the device you installed it on. Simply type in your device's ip address along with the port Nessus uses, 8834.

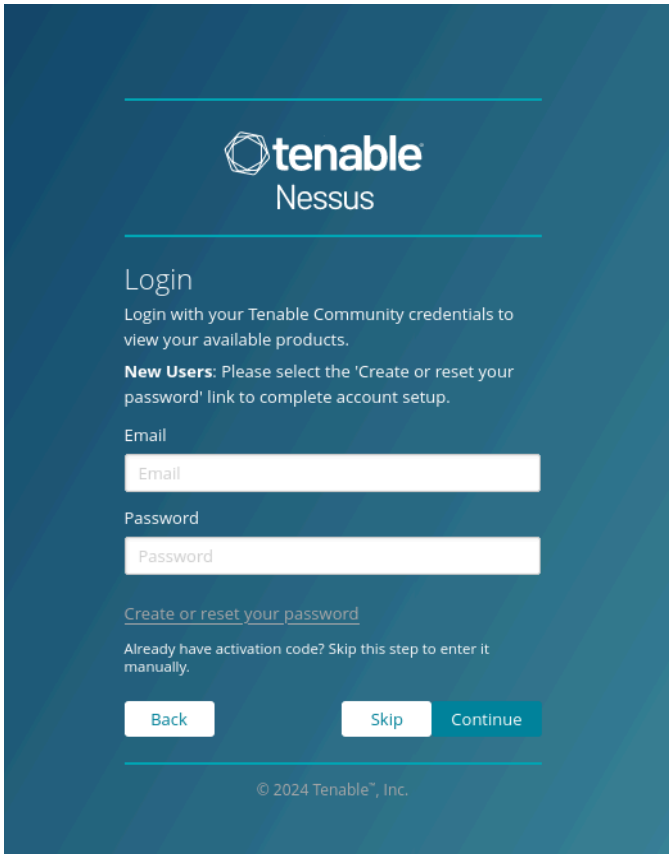


Section #3 Configuring and Initializing Nessus

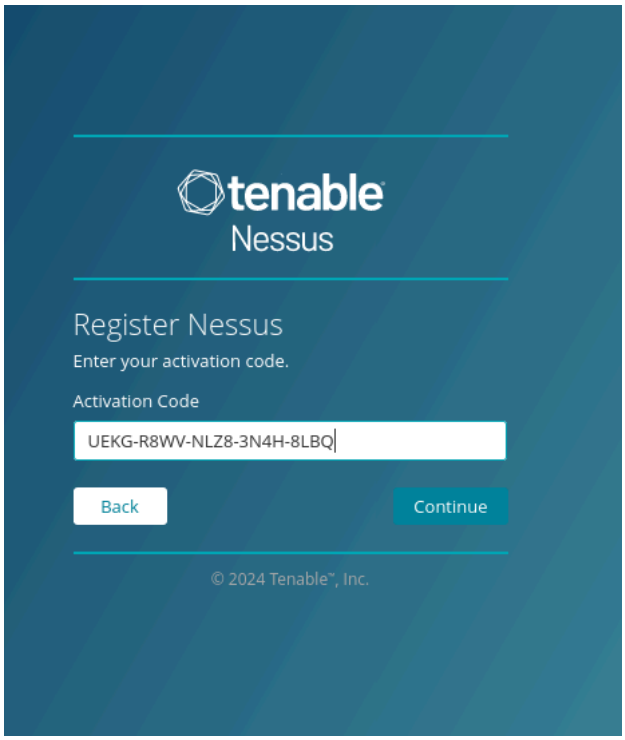
3.1 Choose register for Nessus Essentials:

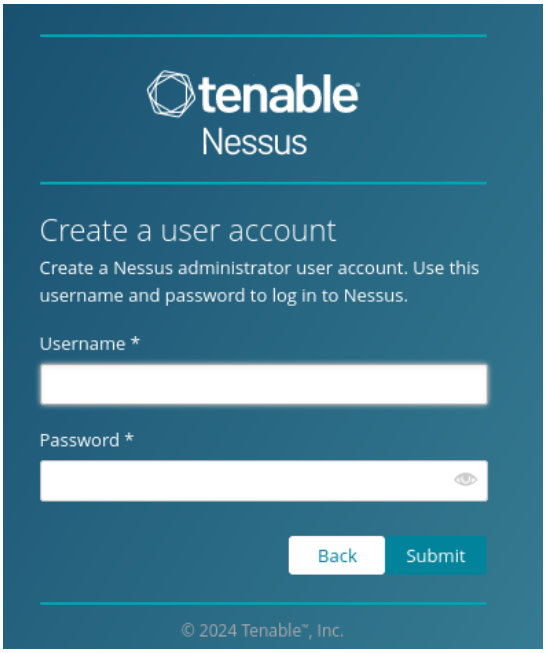


3.2. Skip logging in if you already have a license key.

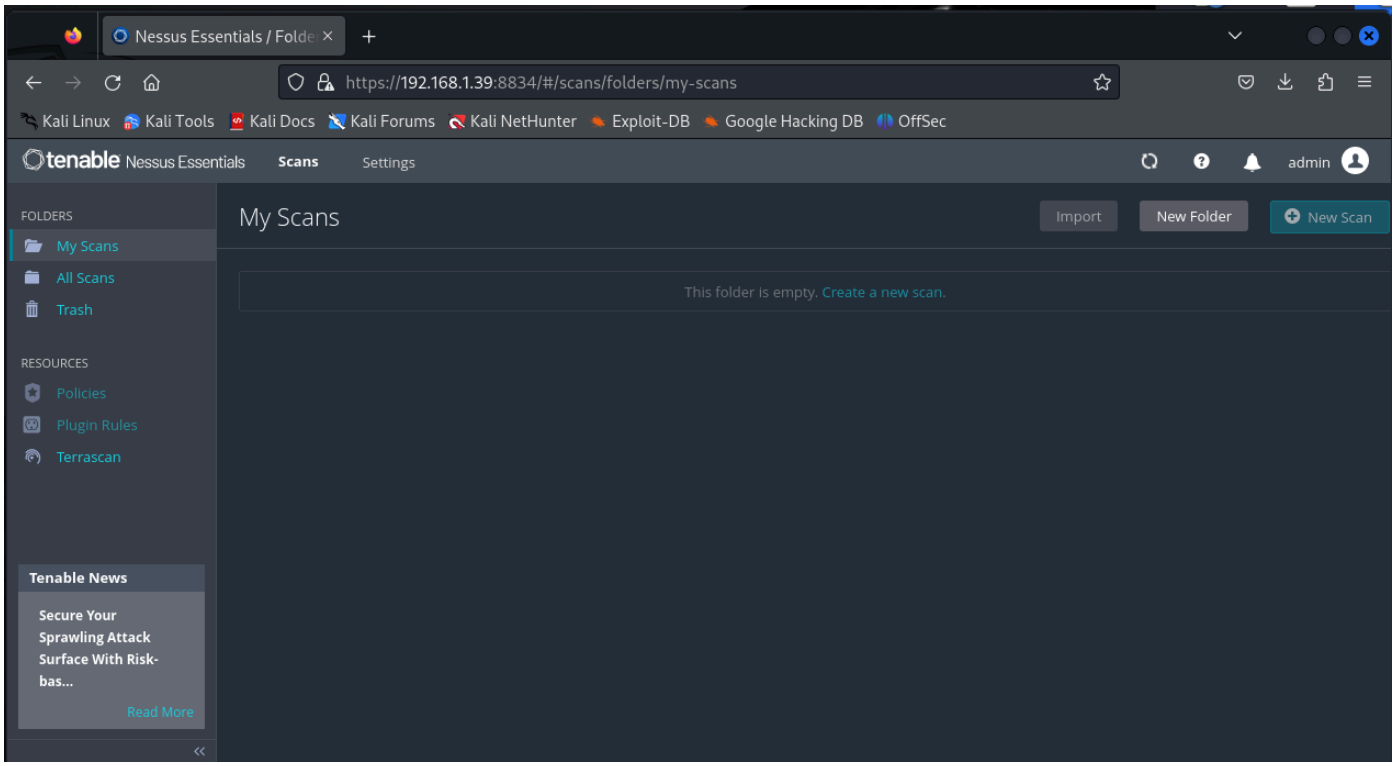


After skipping, enter in the license key, then create your login info for the Nessus Essentials scanner.






tenable
Nessus
Initializing
Please wait while Nessus is initializing.
Downloading plugins...
© 2024 Tenable™, Inc.


Nessus Essentials / Folde
https://192.168.1.39:8834/#/scans/folders/my-scans
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
tenable Nessus Essentials
Scans
Settings
admin
FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan
Tenable News
Secure Your Sprawling Attack Surface With Risk-bas...
Read More
My Scans
Import
New Folder
New Scan
This folder is empty. Create a new scan.

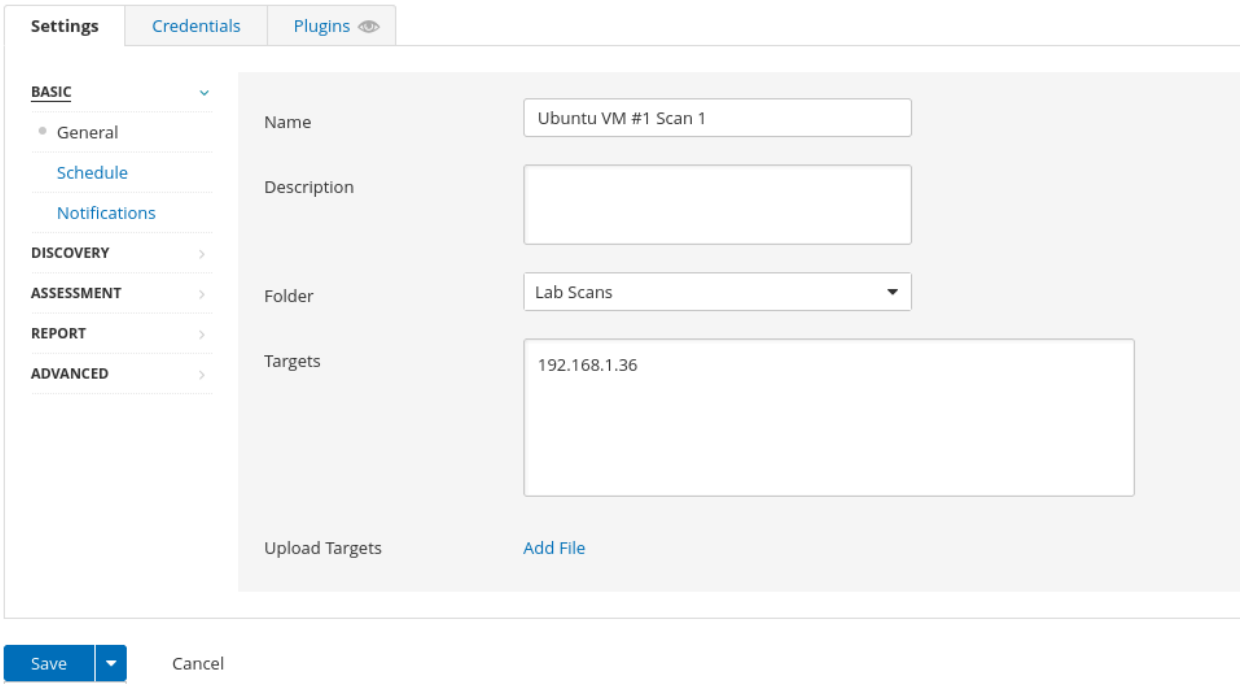
Section #4 Basic Scanning:

A. The first type of scan that I will perform is basic, non credentialed scans. Non-credentialed scans are non-invasive and simulate what an external attacker will see if they were trying to find vulnerabilities on the network devices. Unlike credentialed scans, these scans do not get a deep view of the system configuration of the devices and merely identify open ports and services. Since these scans only look at the devices from the outside they might not find all vulnerabilities, and if vulnerabilities are found they might be false-positives.

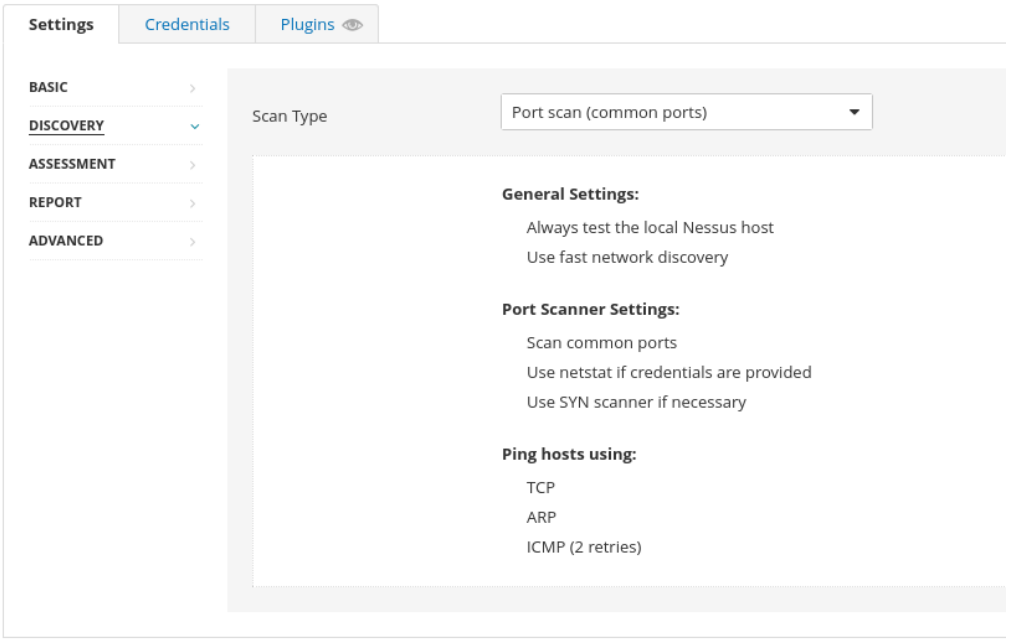
B. Scanning Ubuntu Desktop VM #1 (IPv4 Address 192.168.1.36):

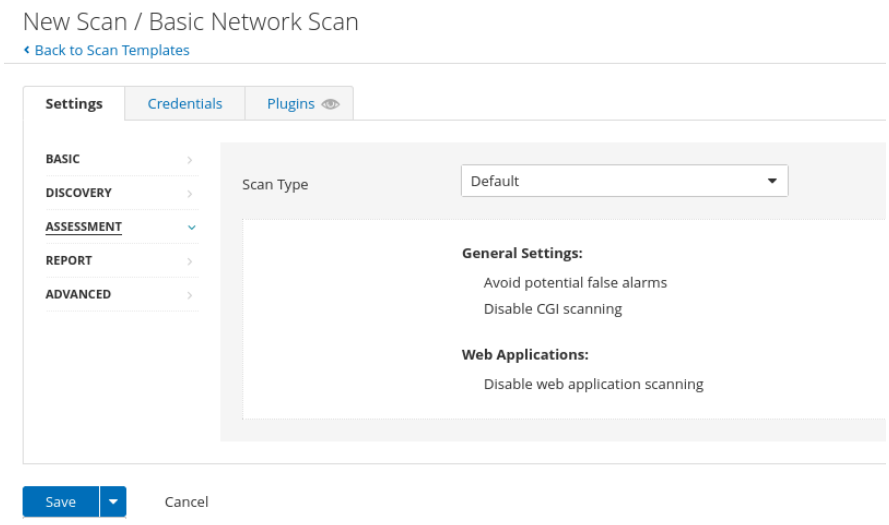
For this first scan I will demonstrate the steps in detail, subsequent scans will omit the details since I will use the same parameters for them except for the name and the ip address.

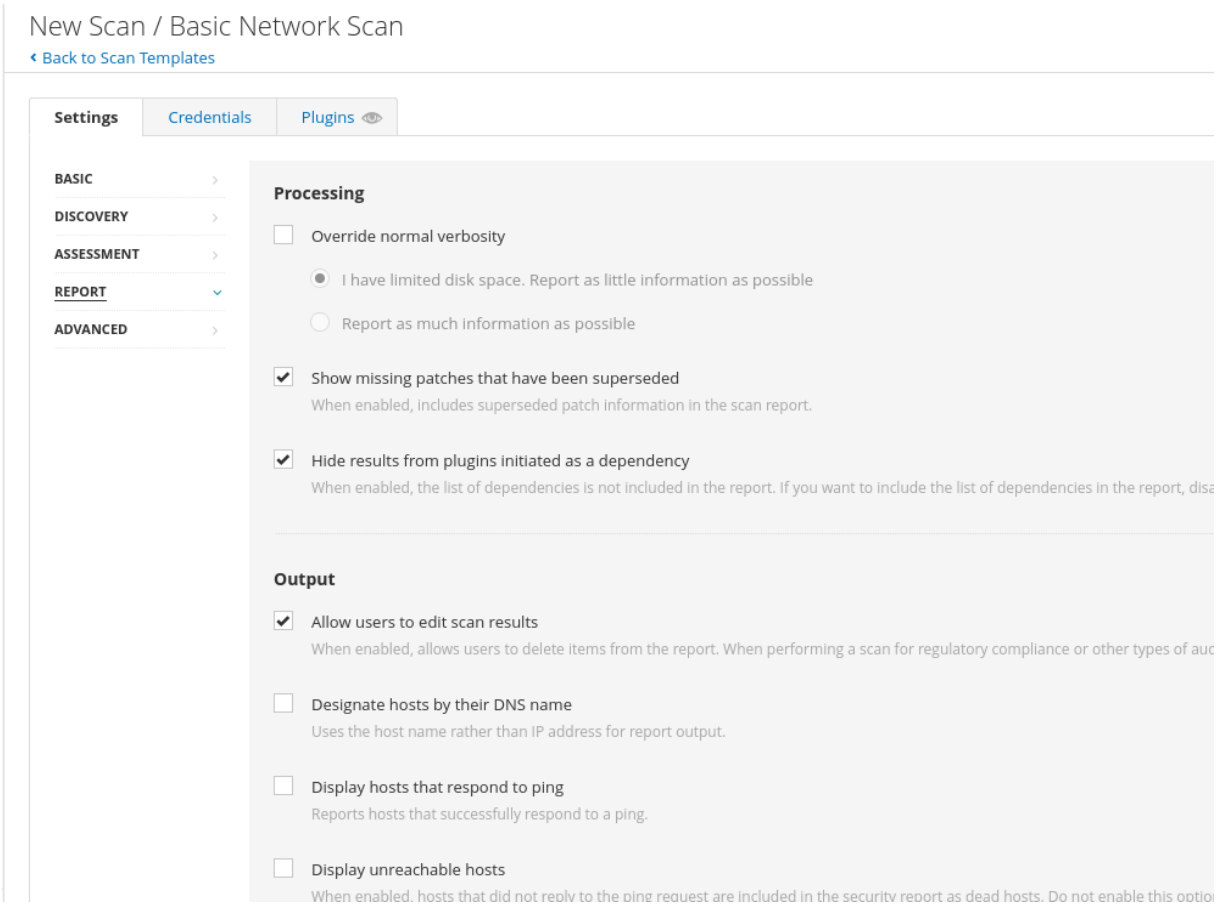
1. Create the scan name and specify the target ip address:



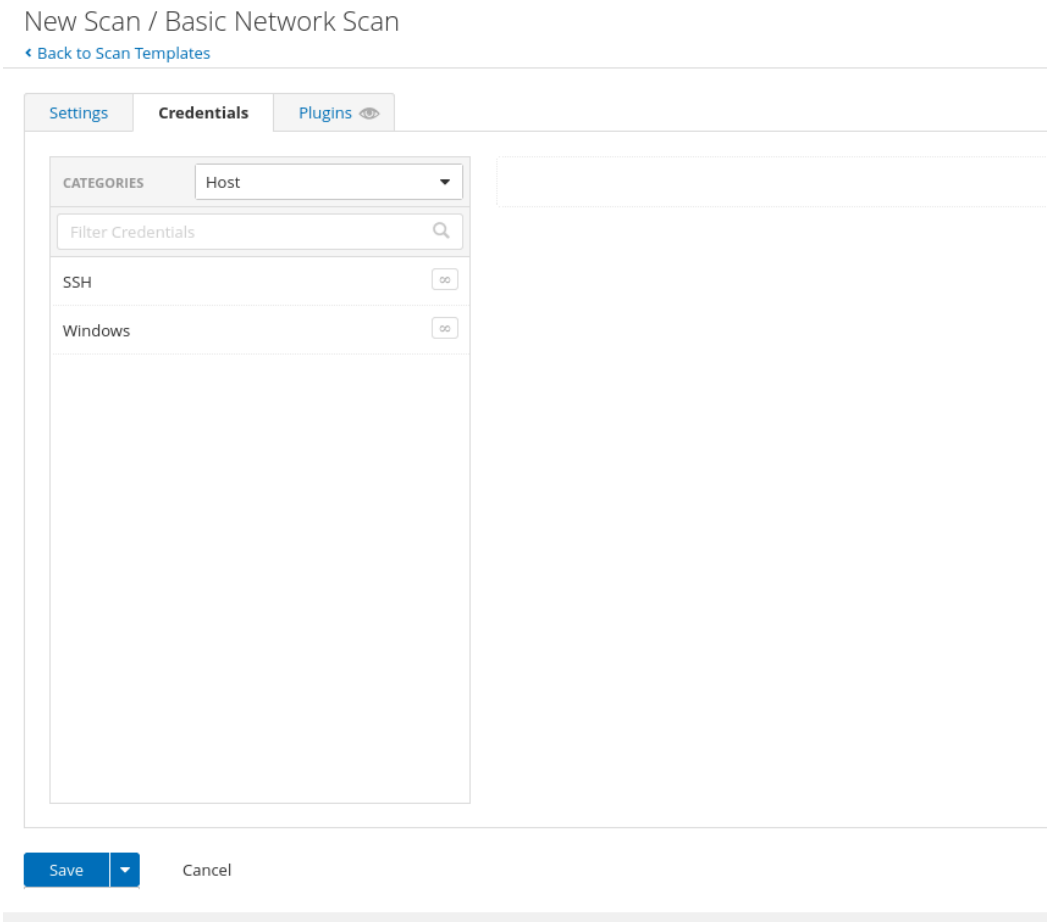
2. Keep all the defaults for the other parameters:


Settings
Credentials
Plugins
BASIC
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Scan Type
Port scan (common ports)
General Settings:
Always test the local Nessus host
Use fast network discovery
Port Scanner Settings:
Scan common ports
Use netstat if credentials are provided
Use SYN scanner if necessary
Ping hosts using:
TCP
ARP
ICMP (2 retries)
Save
Cancel

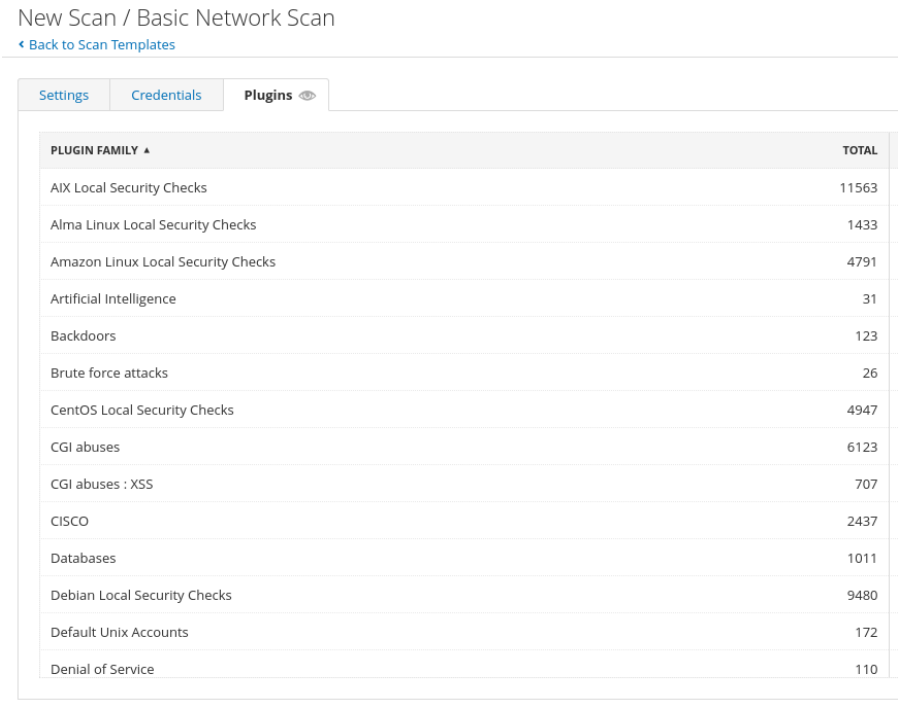

New Scan / Basic Network Scan
Back to Scan Templates
Settings
Credentials
Plugins
BASIC
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Scan Type
Default
General Settings:
Avoid potential false alarms
Disable CGI scanning
Web Applications:
Disable web application scanning
Save
Cancel



3. I won't include credentials this time around:

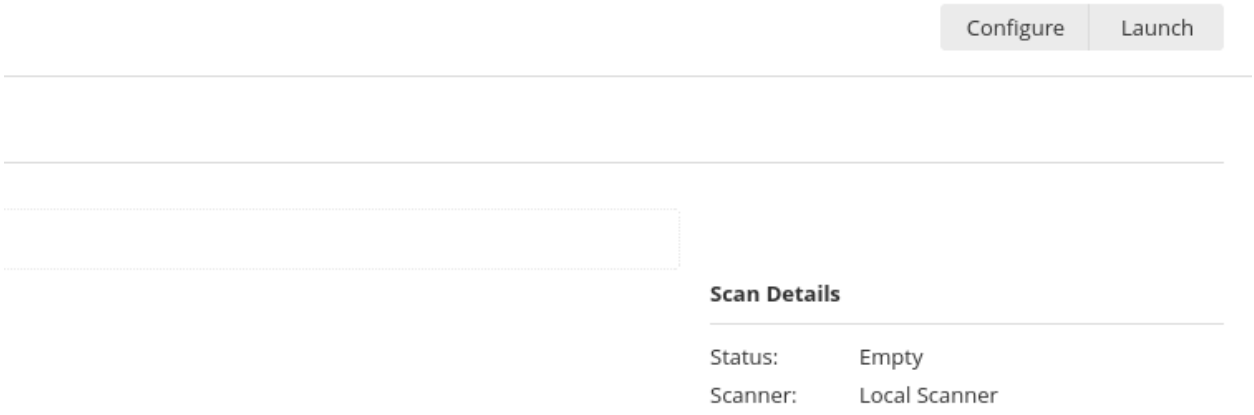


4. I also don't have to do anything for the plugins:

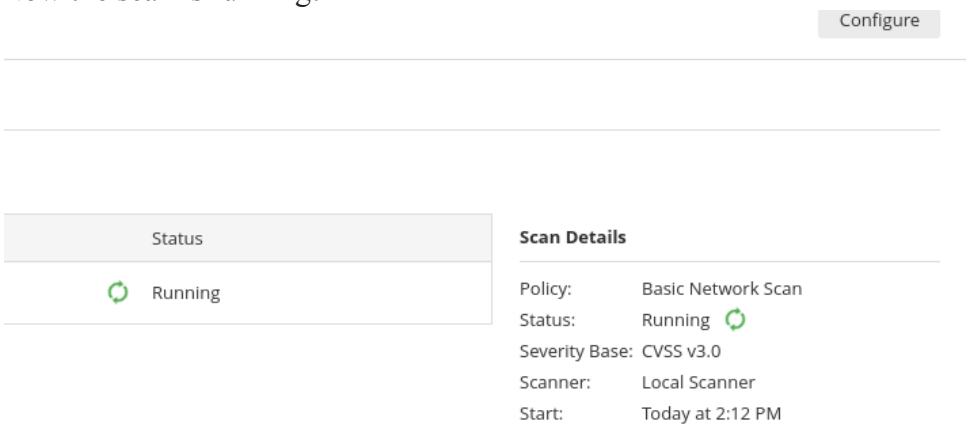


5. Perform the scan:

To perform the scan you have to go to click on the scan you created, then click launch:

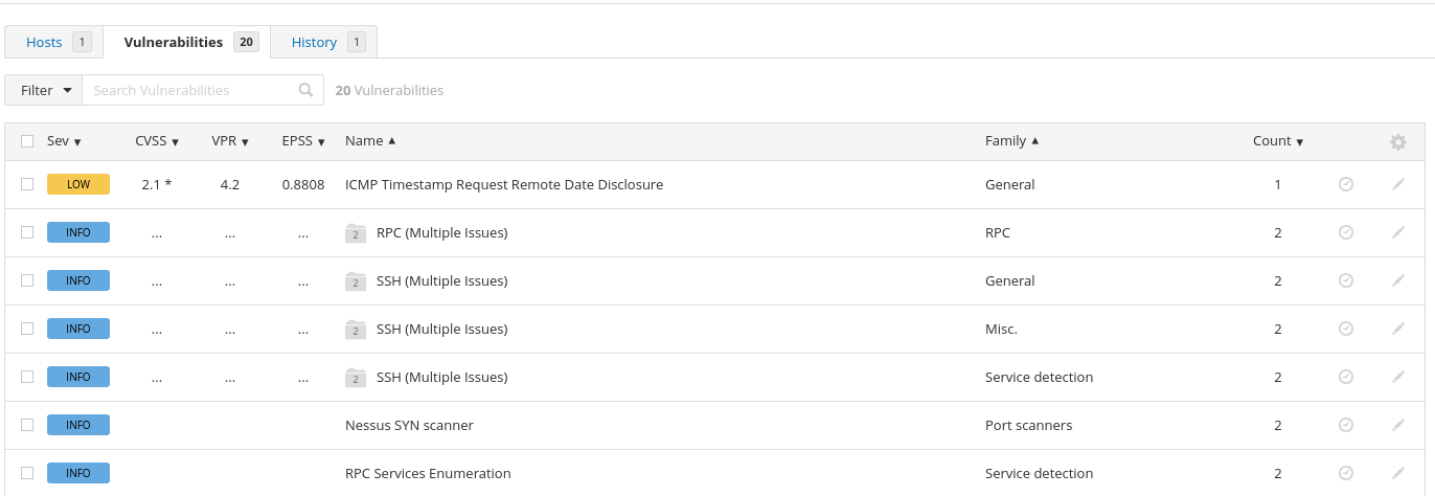


Now the scan is running!

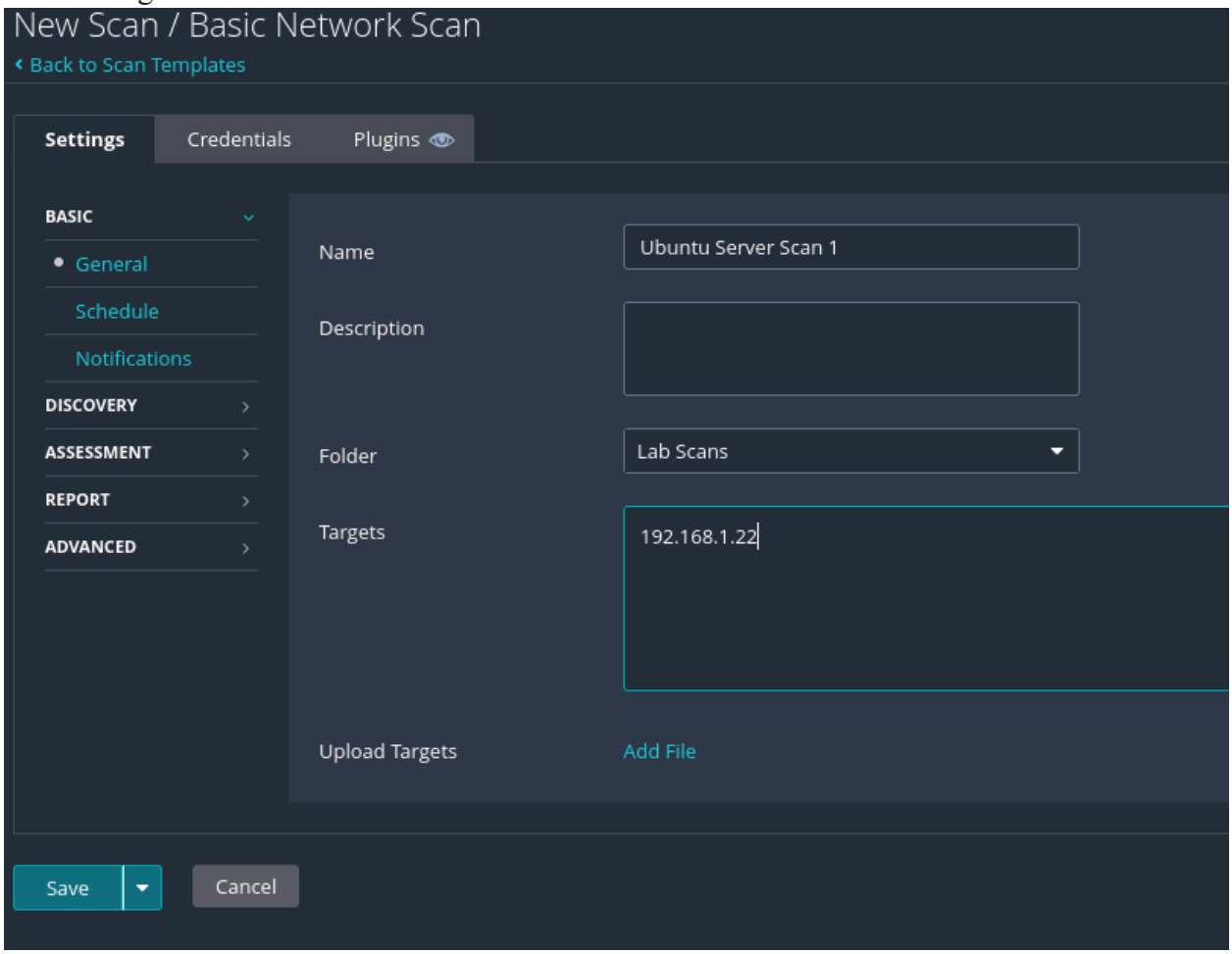


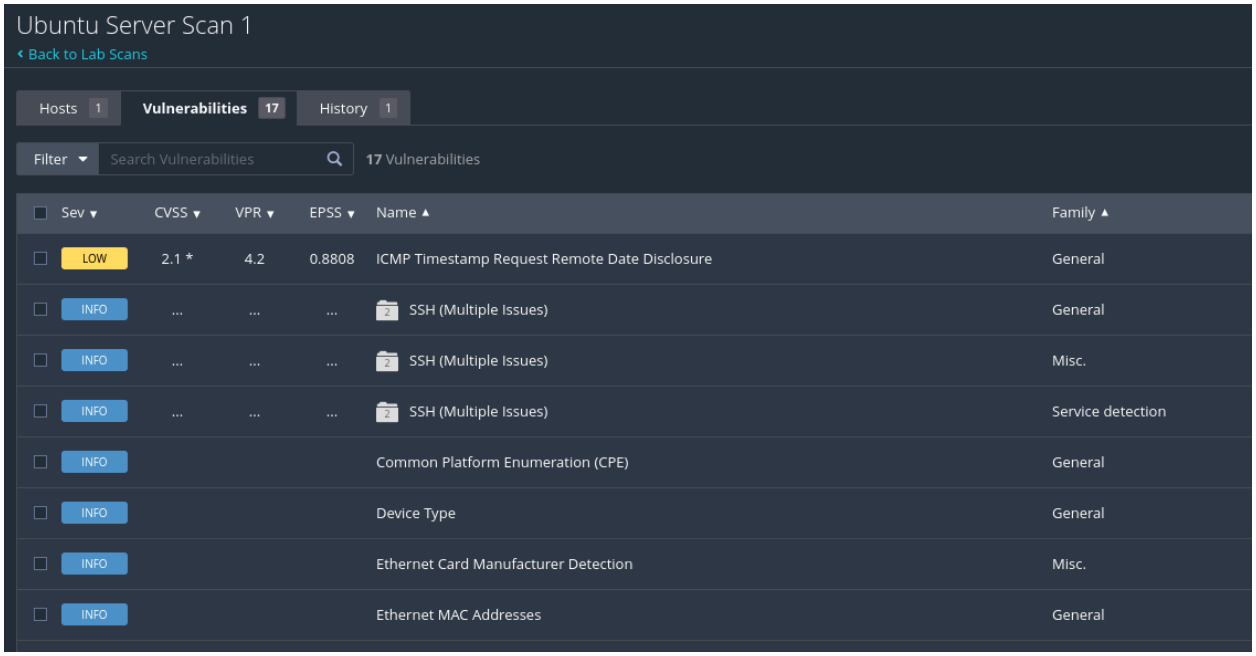
6. Look at the results:

I won’t analyze the results in detail now, but here are the results of the uncredentialed basic scan:



C. Scanning Ubuntu Server VM:



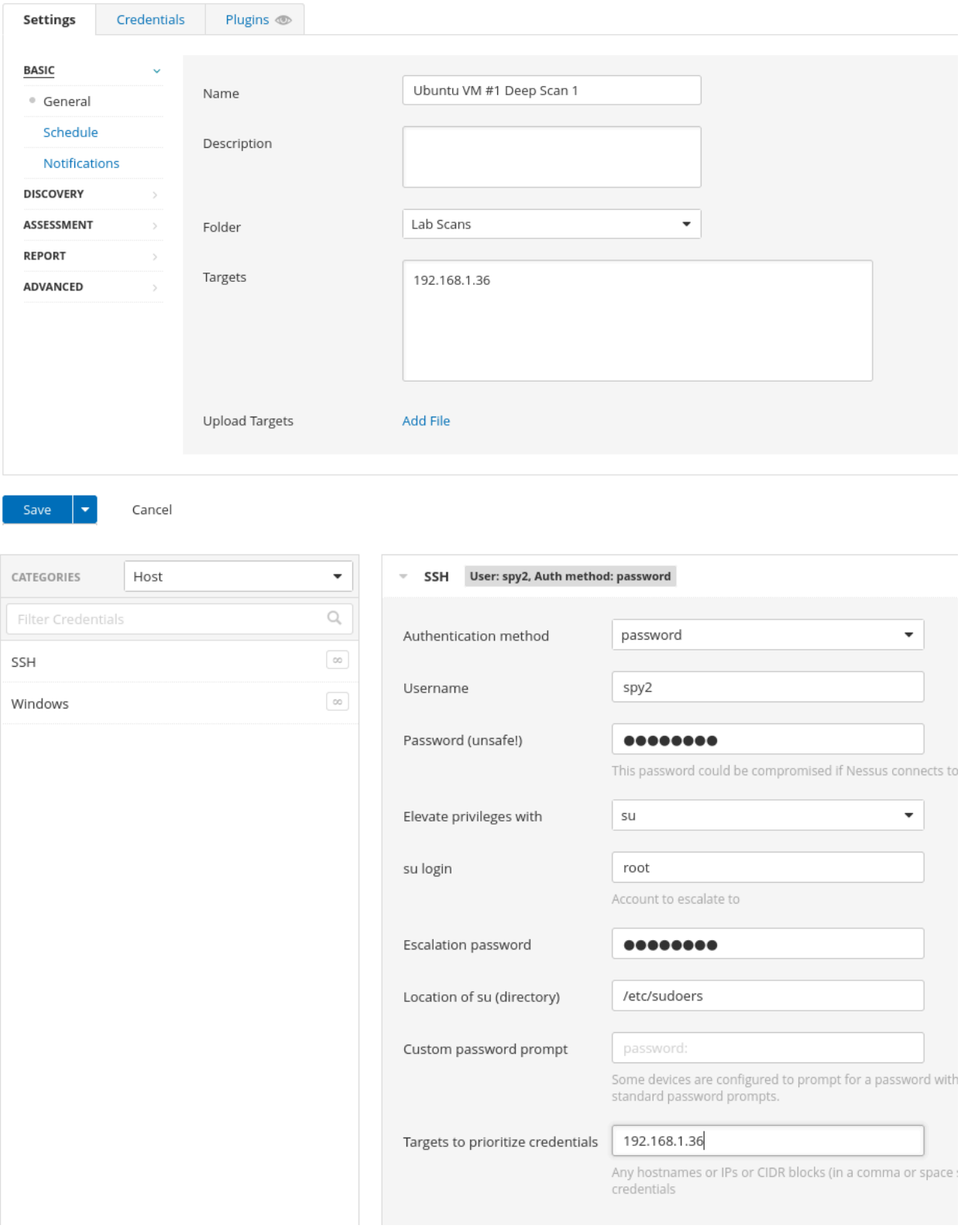


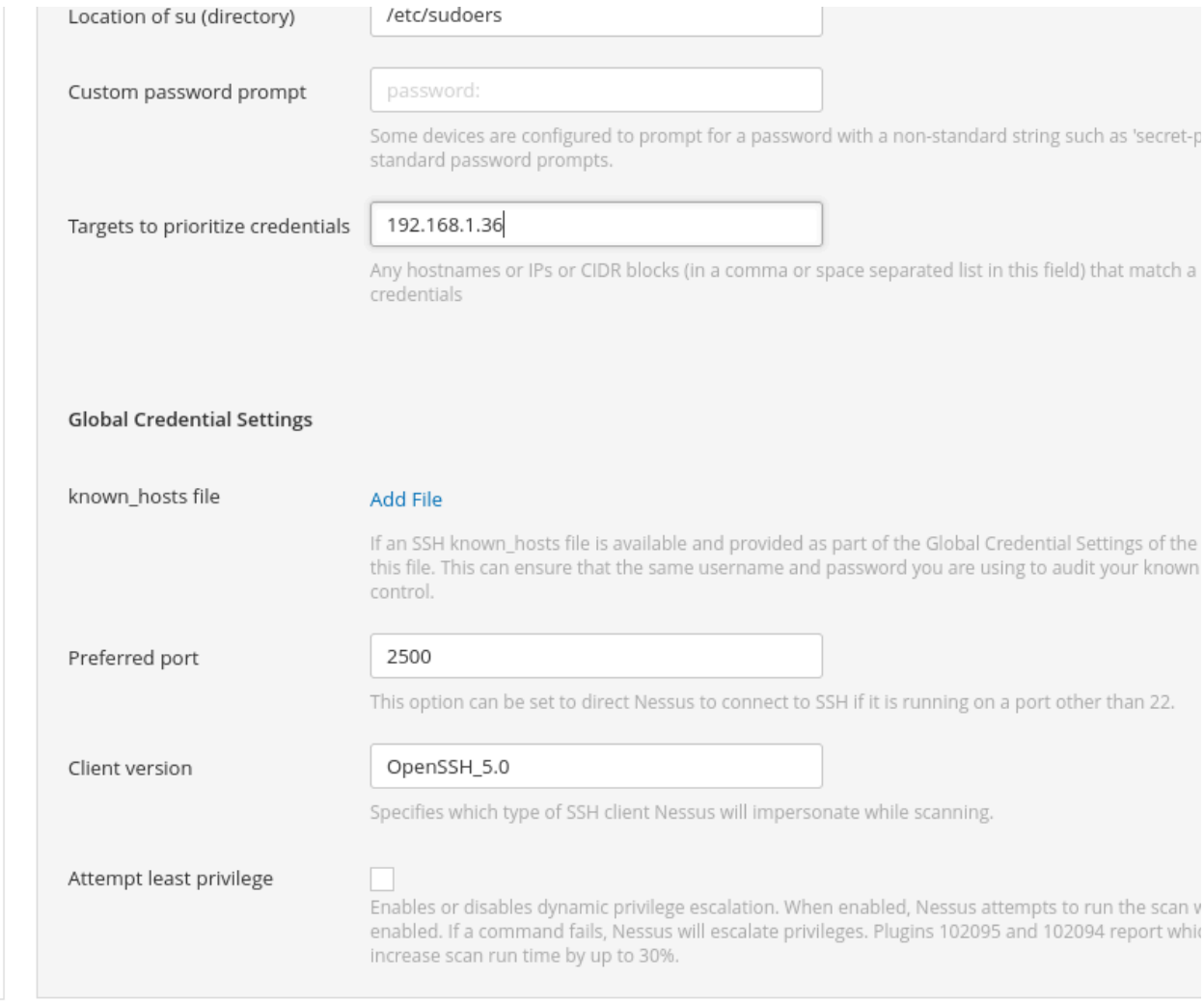
Section #5 Advanced Scanning:

A. The second type of scan I will perform is the credentialed scans. Like I mentioned before these scans can perform a deeper scan of the devices and identify system configuration vulnerabilities unlike non credentialed scans. It is good practice to perform both non-credentialed scans and credentialed scans since they can complement each other. The credentialed scans can find vulnerabilities that the non credentialed scans missed, assuring that all vulnerabilities are identified on the devices. Credentialed scans are also great to utilize in an environment where network security protections like firewalls and intrusion prevention systems are prevalent. Firewalls and IPSs will interfere with regular external, non credentialed scans. With credentialed scans the vulnerability scanner can subvert the network defenses and perform the scan unimpeded.

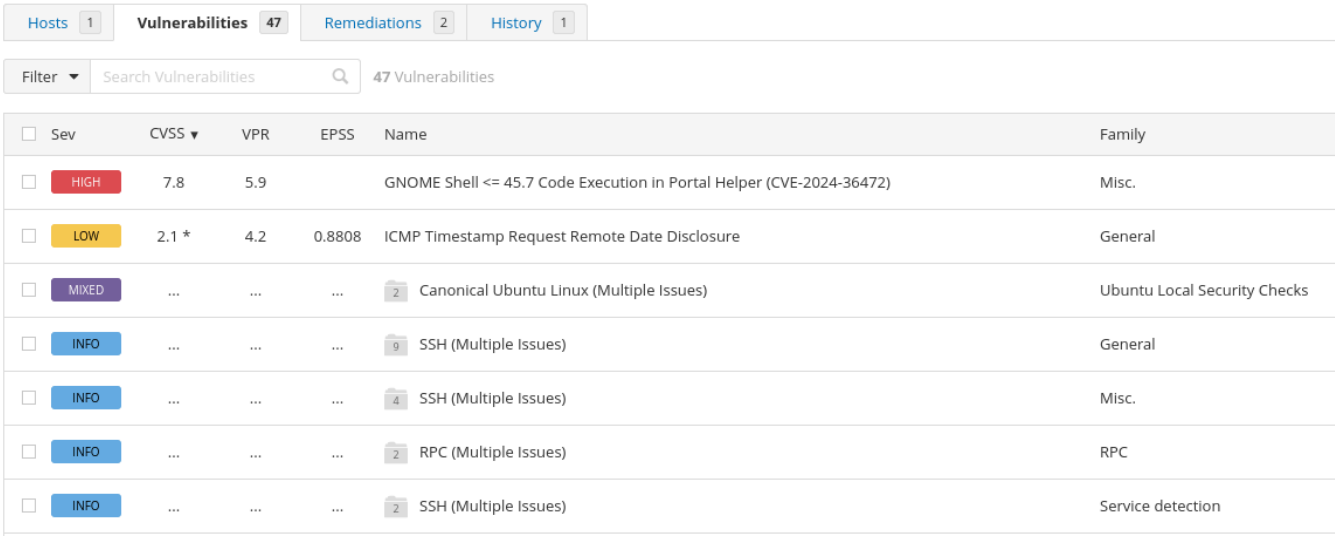
B. Scanning Ubuntu Desktop VM #1 (IPv4 Address 192.168.1.36):

1. I will now create the credentialed scan for the Ubuntu Desktop VM #1. I will use the same parameters as the non credentialed scan, except this time I will provide the credentials for SSH login so the scanner goes deep into the VM.

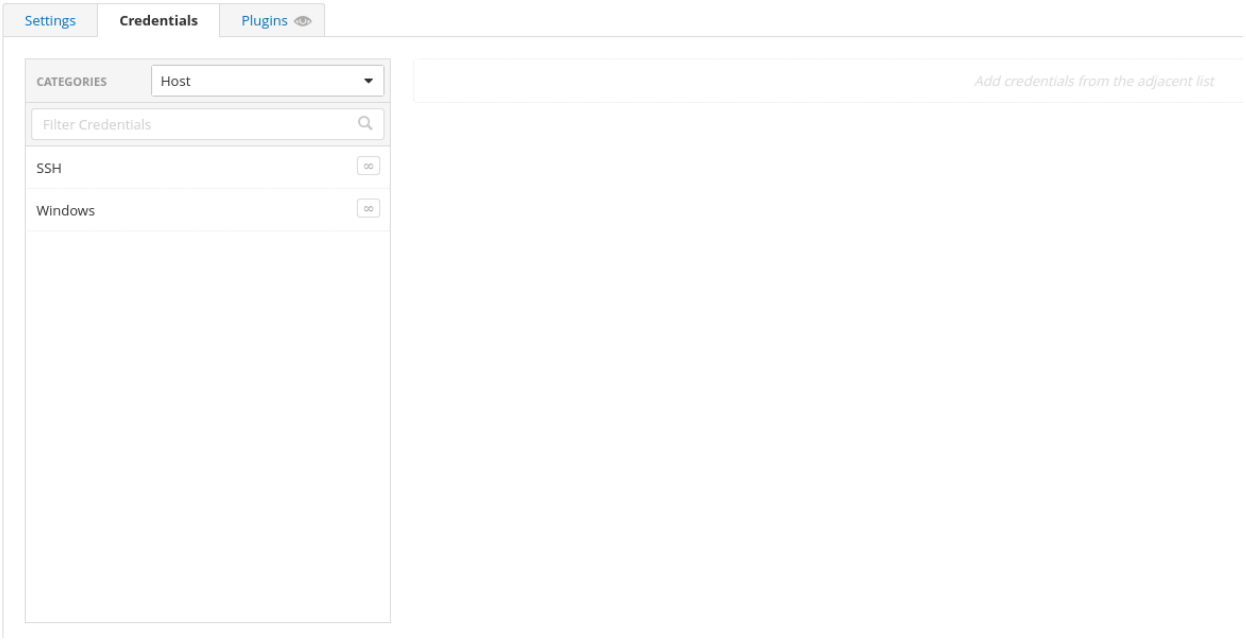

Settings
Credentials
Plugins
BASIC
General
Schedule
Notifications
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Name
Ubuntu VM #1 Deep Scan 1
Description
Folder
Lab Scans
Targets
192.168.1.36
Upload Targets
Add File
Save
Cancel
CATEGORIES
Host
Filter Credentials
SSH
Windows
SSH
User: spy2, Auth method: password
Authentication method
password
Username
spy2
Password (unsafe!)
This password could be compromised if Nessus connects to
Elevate privileges with
su
su login
root
Account to escalate to
Escalation password
Location of su (directory)
/etc/sudoers
Custom password prompt
password:
Some devices are configured to prompt for a password with
standard password prompts.
Targets to prioritize credentials
192.168.1.36
Any hostnames or IPs or CIDR blocks (in a comma or space
credentials



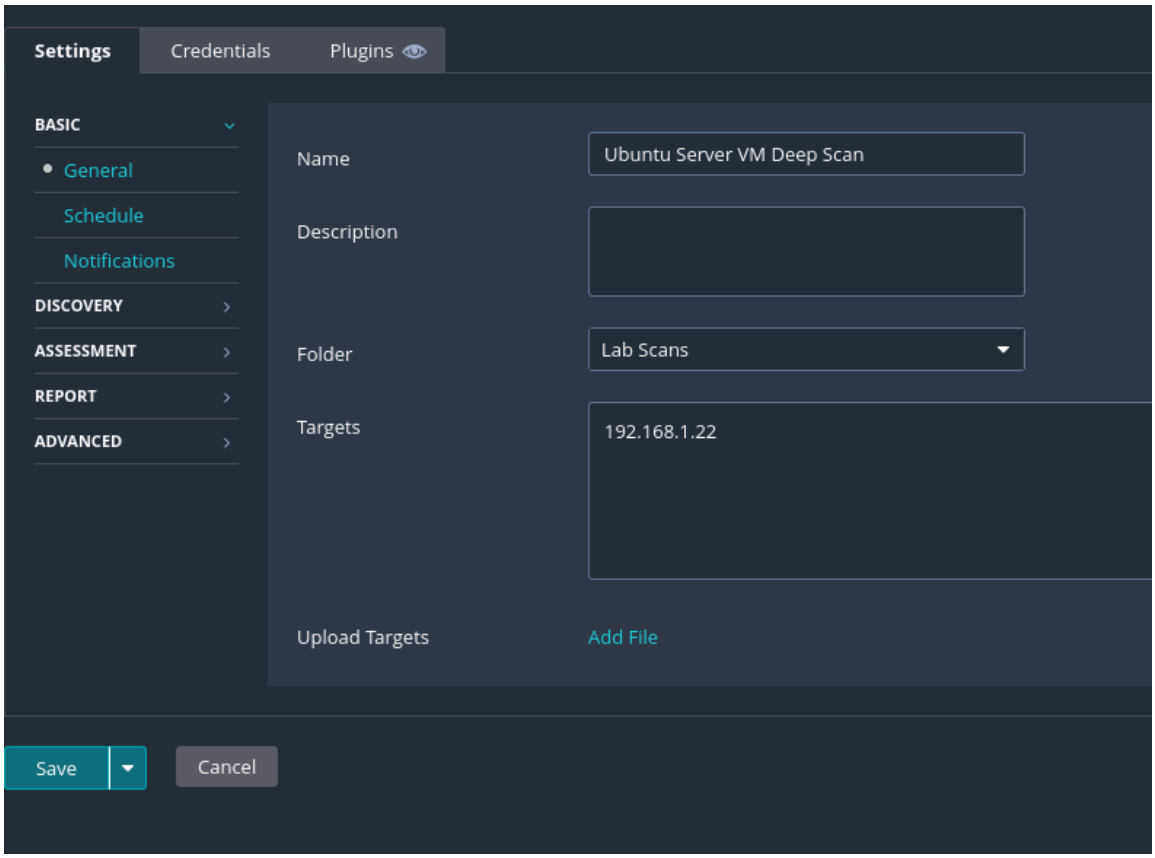
2. Look at results of the credentialed scan:

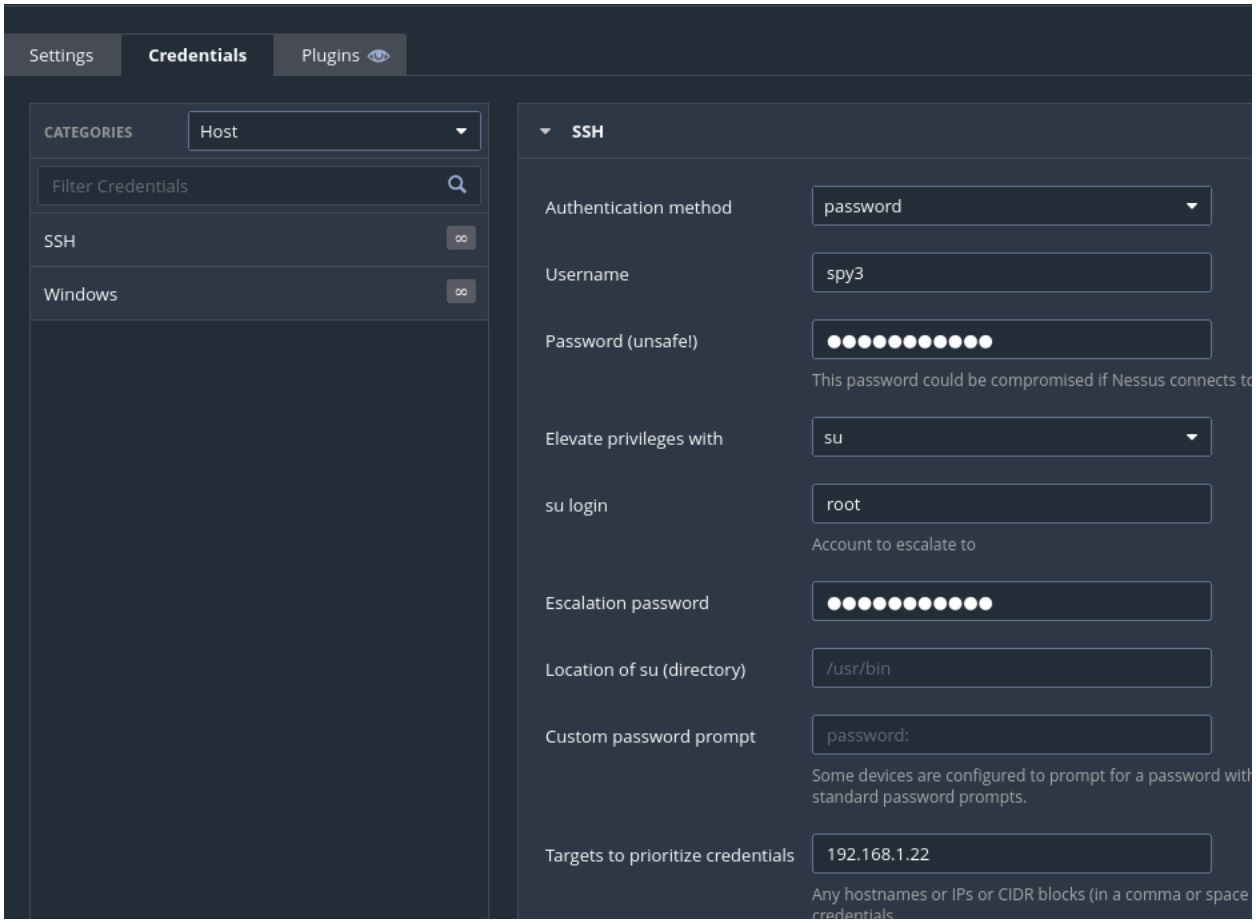


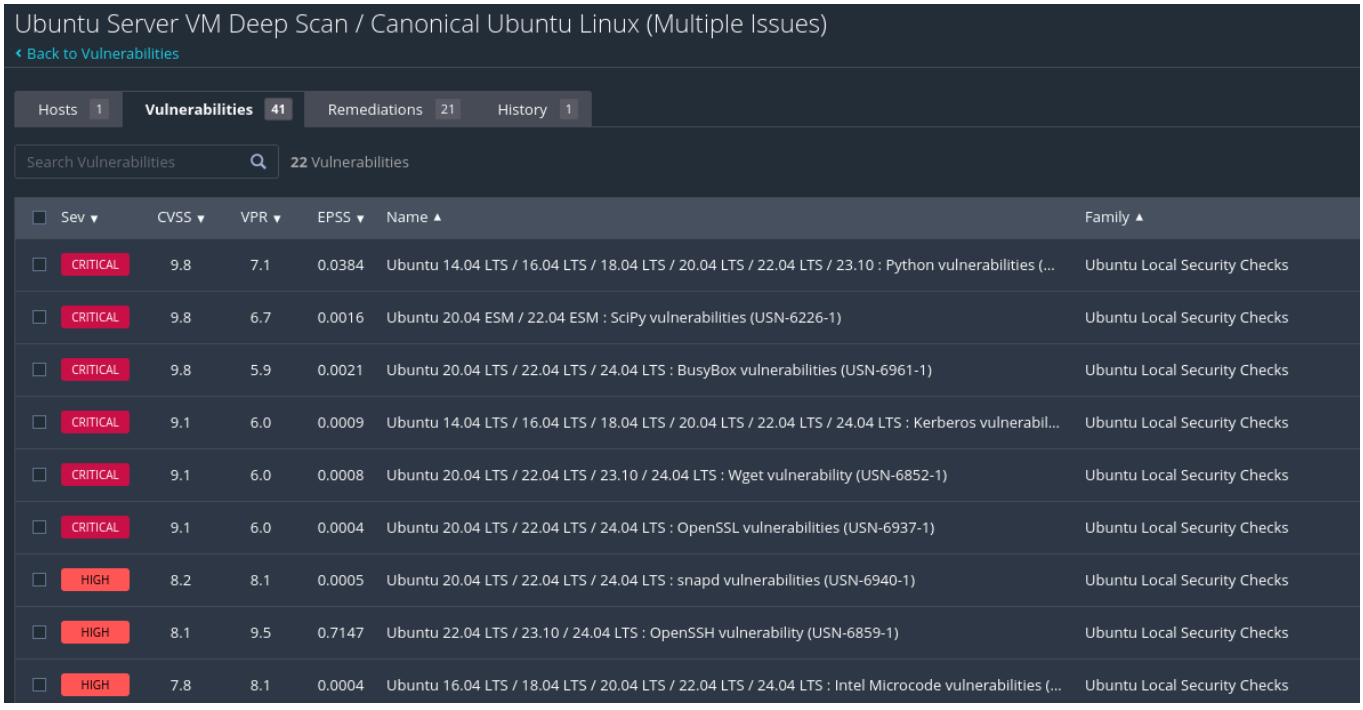
3. The last thing that you should do after the scan is delete the credentials for Nessus:



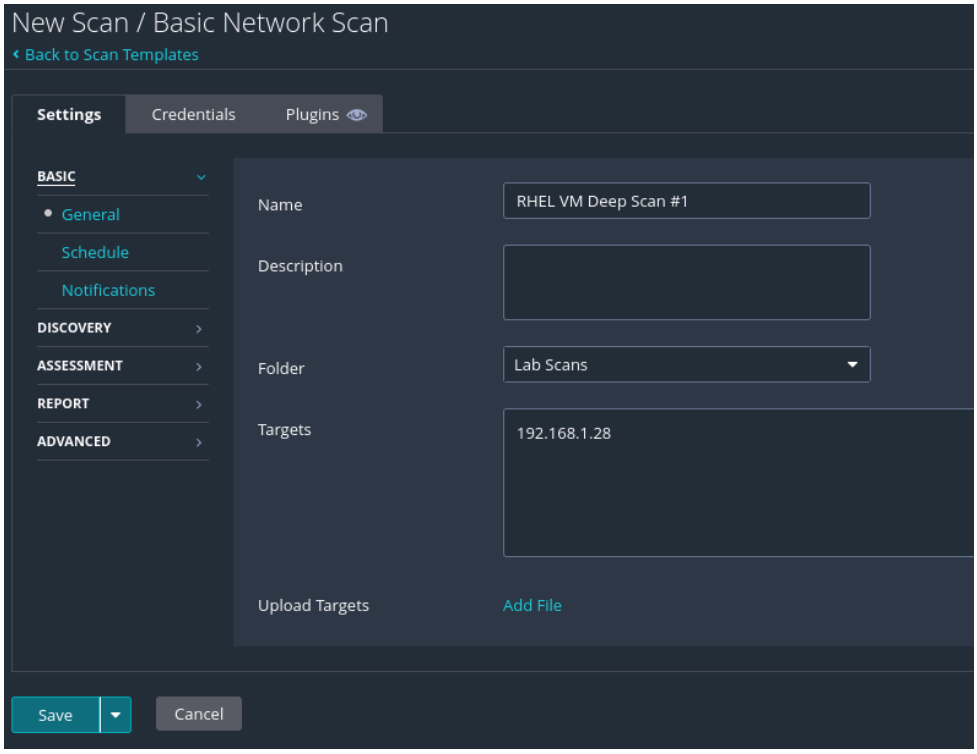
C. Scanning Ubuntu Server VM #2 (IPv4 Address):

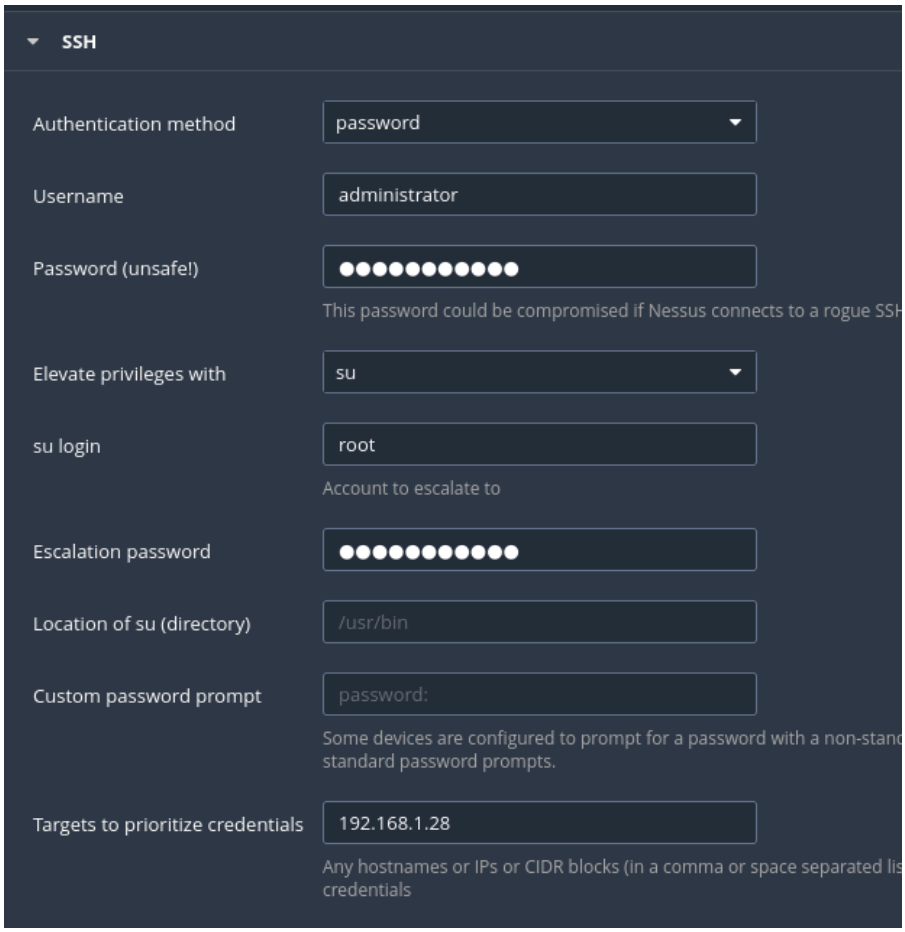


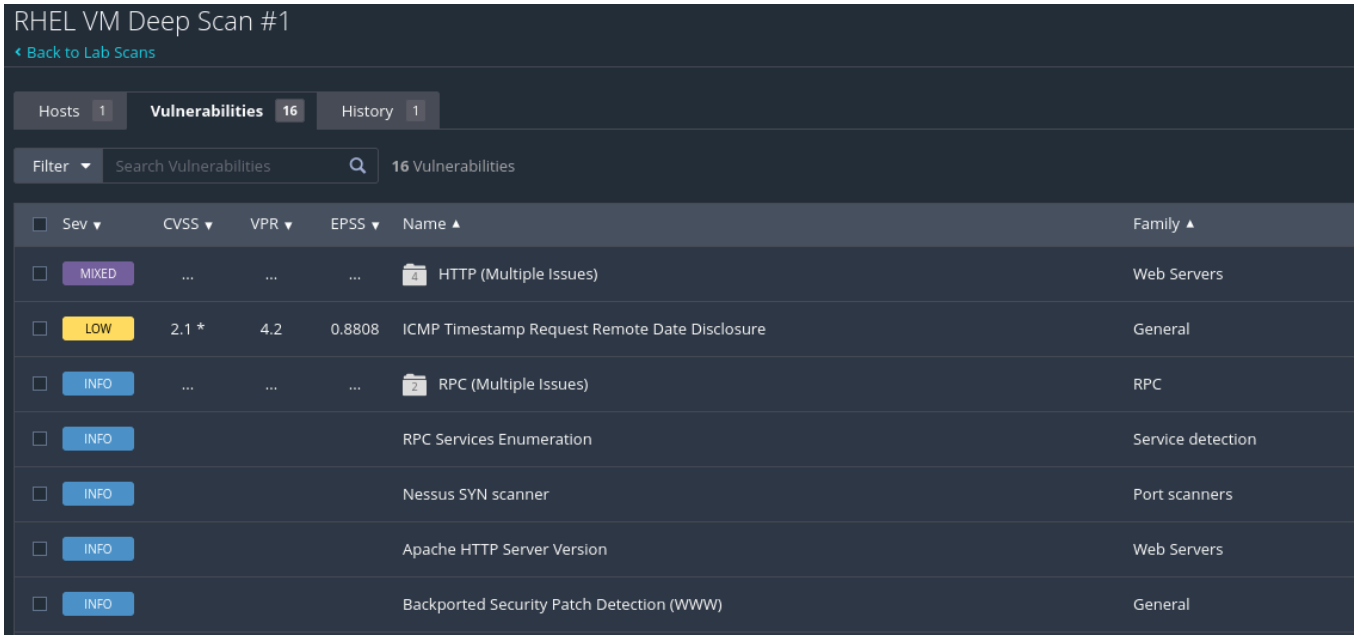




D. Scanning RHEL server VM (IPv4 Address 192.168.1.28):


New Scan / Basic Network Scan
Back to Scan Templates
Settings
Credentials
Plugins
BASIC
General
Schedule
Notifications
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Name
RHEL VM Deep Scan #1
Description
Folder
Lab Scans
Targets
192.168.1.28
Upload Targets
Add File
Save
Cancel


SSH
Authentication method
password
Username
administrator
Password (unsafe!)
This password could be compromised if Nessus connects to a rogue SSH
Elevate privileges with
su
su login
root
Account to escalate to
Escalation password
Location of su (directory)
/usr/bin
Custom password prompt
password:
Some devices are configured to prompt for a password with a non-stand
standard password prompts.
Targets to prioritize credentials
192.168.1.28
Any hostnames or IPs or CIDR blocks (in a comma or space separated lis
credentials

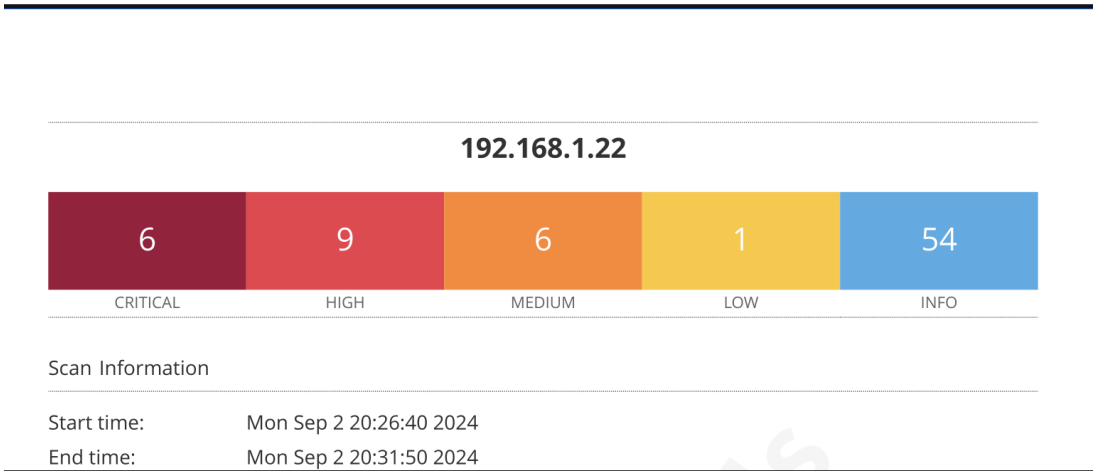


Section #6 Interpreting Scan Results:

For the purposes of this lab I will only be interpreting the results based on the credentialed scans because these scans provide much more information than the non credentialed scans. I will only interpret the results for the Ubuntu Server VM, and focus on critical vulnerabilities only to save time as well. Also the scans used the CVSS 3.0 scoring system even though it has been replaced with CVSS 4.0.

A. Interpret the Scan Report for Ubuntu Server VM :

1. Create a PDF report of the scan:



I created a PDF report for the Ubuntu Server VM scan. The report contains details for all the vulnerabilities found such as the cvss score, how the vulnerability works, and some remediation steps to fix it. Nessus is helpful since the remediation information is highly detailed and straightforward to implement.

2. Interpret the results for Ubuntu Server VM:

The Ubuntu Server VM has 6 critical vulnerabilities and 9 high.

1. The first critical vulnerability Nessus found is actually multiple CVEs related to Python.

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Integrity
Score: 9.8	Network	Low	None	None	Unchanged	High	High	High

2. The second critical vulnerability is related to Kerberos which is a network authentication protocol used to control access to network resources. It should be noted that I do not use Kerberos in my lab environment

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Availability
Score: 9.1	Network	Low	None	None	Unchanged	High	None	High

3. The third critical vulnerability is related to the SciPy package.

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Availability
Score: 9.8	Network	Low	None	None	Unchanged	High	High	High

4. Critical vulnerability four is related to the wget package.

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Availability
Score: 9.1	Network	Low	None	None	Unchanged	High	High	None

5. Critical vulnerability 5 is related to the BusyBox package.

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Availability
Score: 9.8	Network	Low	None	None	Unchanged	High	High	High

6. Critical Vulnerability 6 is related to the OpenSSL package.

CVSS 3.0 Score	Attack Vector	Attack Complexity	Privileges Required	User Interaction	Scope	Confidentiality	Integrity	Availability
Score: 9.1	Network	Low	None	None	Unchanged	High	None	High

7. Choose the most urgent vulnerabilities to remediate for the next section:

I will make my decisions regarding the order for remediation based on a few factors: 1. The CVSS base score, 2. Is the full CIA triad compromised by the vulnerability?, and lastly 3. Is the remediation easy?

Vulnerability	CVSS Base Score	Full CIA compromised?	Remediation easy?
Python Packages	9.8	Yes	Yes (update packages)

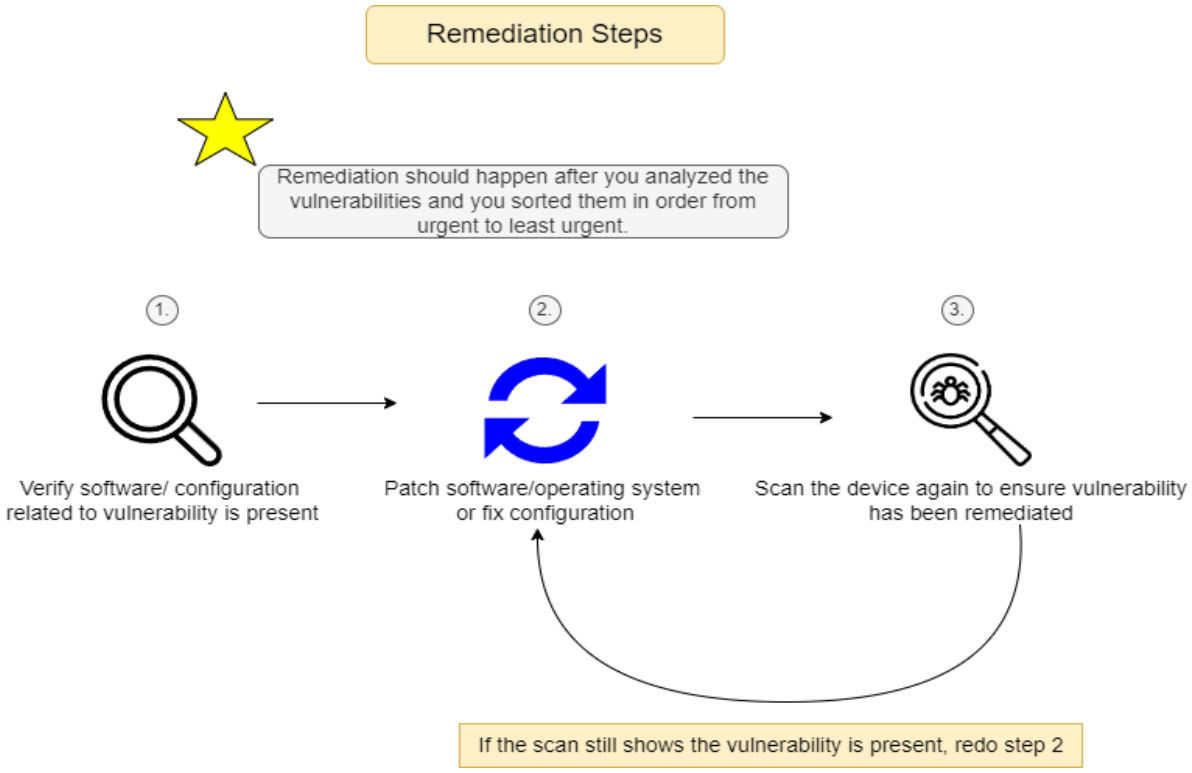
Kerberos	9.1	No	Yes (update packages)
SciPy	9.8	Yes	No (Requires Ubuntu Pro)
Wget	9.1	No	Yes (update packages)
BusyBox	9.8	Yes	Yes (update packages)
OpenSSL	9.1	No	Yes (update packages)

So, I will do the Python vulnerability first, then BusyBox. I will skip SciPy since I do not have Ubuntu Pro, and I do not use it. Next I will do the Kerberos, Wget, and OpenSSL patches.

Section #6 Remediation Strategies:

Note: I will only demonstrate the remediation strategies I implemented for the Ubuntu Server VM to cut the size of this lab writeup down. Also, I will only demonstrate the steps I took to remediate the critical vulnerabilities since they are the most urgent ones that must be remediated.

Remediation Strategy:



6a. Remediate Ubuntu Server Vulnerabilities:

1. Verify the vulnerabilities are present and patch if present:

   To verify the vulnerabilities are present I will create a bash script that will automate the process. The bash script will take the package name, the version needed, and if the package version needed to remediate the vulnerability is not present, the script will update the package.

Package	Version needed for remediation

Python	**Solution** Update the affected packages. **See Also** https://ubuntu.com/security/notices/USN-6891-1 **Output** - Installed package : libpython3.10_3.10.12-1~22.04.3 - Fixed package : libpython3.10_3.10.12-1~22.04.4 - Installed package : libpython3.10-dev_3.10.12-1~22.04.3 - Fixed package : libpython3.10-dev_3.10.12-1~22.04.4 - Installed package : libpython3.10-minimal_3.10.12-1~22.04.3 - Fixed package : libpython3.10-minimal_3.10.12-1~22.04.4 - Installed package : libpython3.10-stdlib_3.10.12-1~22.04.3 - Fixed package : libpython3.10-stdlib_3.10.12-1~22.04.4 - Installed package : python3.10_3.10.12-1~22.04.3 - Fixed package : python3.10_3.10.12-1~22.04.4 - Installed package : python3.10-dev_3.10.12-1~22.04.3 - Fixed package : python3.10-dev_3.10.12-1~22.04.4 - Installed package : python3.10-minimal_3.10.12-1~22.04.3 - Fixed package : python3.10-minimal_3.10.12-1~22.04.4
SciPy	**Solution** Update the affected python3-scipy package. **See Also** https://ubuntu.com/security/notices/USN-6226-1 **Output** NOTE: This vulnerability check contains fixes that apply to packages only available in Ubuntu ESM repositories. Access to these package security updates require an Ubuntu Pro subscription. - Installed package : python3-scipy_1.8.0-1exp2ubuntu1 - Fixed package : python3-scipy_1.8.0-1exp2ubuntu1+esm1
BusyBox	**Solution** Update the affected packages. **See Also** https://ubuntu.com/security/notices/USN-6961-1 **Output** - Installed package : busybox-initramfs_1:1.30.1-7ubuntu3 - Fixed package : busybox-initramfs_1:1.30.1-7ubuntu3.1 - Installed package : busybox-static_1:1.30.1-7ubuntu3 - Fixed package : busybox-static_1:1.30.1-7ubuntu3.1

Kerberos	**Solution** Update the affected packages. **See Also** https://ubuntu.com/security/notices/USN-6947-1 **Output** - Installed package : libgssapi-krb5-2_1.19.2-2ubuntu0.3 - Fixed package : libgssapi-krb5-2_1.19.2-2ubuntu0.4 - Installed package : libk5crypto3_1.19.2-2ubuntu0.3 - Fixed package : libk5crypto3_1.19.2-2ubuntu0.4 - Installed package : libkrb5-3_1.19.2-2ubuntu0.3 - Fixed package : libkrb5-3_1.19.2-2ubuntu0.4 - Installed package : libkrb5support0_1.19.2-2ubuntu0.3 - Fixed package : libkrb5support0_1.19.2-2ubuntu0.4
Wget	**Solution** Update the affected wget package. **See Also** https://ubuntu.com/security/notices/USN-6852-1 **Output** - Installed package : wget_1.21.2-2ubuntu1 - Fixed package : wget_1.21.2-2ubuntu1.1
OpenSSL	**Solution** Update the affected packages. **See Also** https://ubuntu.com/security/notices/USN-6937-1 **Output** - Installed package : libssl3_3.0.2-0ubuntu1.15 - Fixed package : libssl3_3.0.2-0ubuntu1.17 - Installed package : openssl_3.0.2-0ubuntu1.15 - Fixed package : openssl_3.0.2-0ubuntu1.17

Script:

While you can simply use apt upgrade to patch the packages, using my script is a better option for me. Using apt upgrade will update all packages that can be updated, but you

don’t have control over the version installed when doing so. I created the bash script since I want to have control over the package version being installed. I also want the ability to check that either the currently installed package version is the one with the patch or not.

How does the script work?

1. The script prompts users for the number of packages they want to search for, and stores that value in the variable num_packages.
2. Next, using a for loop structure, the script iterates through the prompts based on the num_packages value. For each package the user is prompted to enter the string value for the package name and the version required for remediation.
3. The package_version value is used to create a new variable called installed_version that stores the output of the dpkg -s command used on package_name. The output is also filtered so that only the string value of the version installed is displayed.

```
  GNU nano 6.2                              vuln_remediation.sh *
#!/bin/bash

# This script can be used to  automate the  vulnerability remediation process
# on Ubuntu or other Debian based Linux systems.

# Allow user to choose how many vulnerabilities they want to search for.

   echo "Enter the number of packages you want to search for:"
   read  num_packages

  for ((i=1; i<=num_packages; i++))
  do
# User will enter the package name and the version needed for remediation
    echo "Enter the name for package $i:"
    read  package_name

    echo "Enter the version for package $i needed for remediation:"
    read  package_version

    installed_version=$(dpkg -s $package_name | awk '/^Version:/ {print $2}')

    if dpkg --compare-versions  "$installed_version"  "lt" "$package_version"; then

     echo "The installed version is $installed_version when remediation requires $package_version!"
     echo "Would you like to update the package to remediate it? (yes/no)"
     read response
       if [ "$response" = "yes" ]; then
         sudo apt-get update > /dev/null
         sudo apt-get install $package_name > /dev/null
         echo " The new version is now $installed_version !"
       else
         echo "Thank You. Bye!"
       fi
```

Fix The Vulnerabilities using my script:

1.BusyBox

Something interesting occurred! The currently installed version is the version needed to fix the vulnerability discovered by Nessus. This could possibly be a sign that the vulnerability was a false positive. A false positive means that the vulnerability scanner, in this case Nessus, thought that it found a vulnerability when in fact none was present. This also demonstrates why my script is useful for the remediation process. If I simply used apt upgrade and assumed that all the packages were fixed I would have skipped an important part of the remediation process, verifying the vulnerability is present! I would have updated a package for no reason!

```
spy3@spyserver1:~/Scripts$ sudo ./vuln_remediation.sh
[sudo] password for spy3:
Enter the number of packages you want to search for:
2
Enter the name for package 1:
busybox-initramfs
Enter the version for package 1 needed for remediation:
1:1.30.1-7ubuntu3.1
The installed version is okay since it is version 1:1.30.1-7ubuntu3.1.
Enter the name for package 2:
busybox-static
Enter the version for package 2 needed for remediation:
1:1.30.1-7ubuntu3.1
The installed version is okay since it is version 1:1.30.1-7ubuntu3.1.
spy3@spyserver1:~/Scripts$
```

2. Python Vulnerabilities
False Positive!

```
spy3@spyserver1:~/Scripts$ sudo ./vuln_remediation.sh
Enter the number of packages you want to search for:
7
Enter the name for package 1:
libpython3.10
Enter the version for package 1 needed for remediation:
3.10.12-1~22.04.4
The installed version is okay since it is version 3.10.12-1~22.04.5.
Enter the name for package 2:
libpython3.10-dev
Enter the version for package 2 needed for remediation:
3.10.12-1~22.04.4
The installed version is okay since it is version 3.10.12-1~22.04.5.
Enter the name for package 3:
libpython3.10-minimal
Enter the version for package 3 needed for remediation:
3.10.12-1~22.04.4
The installed version is okay since it is version 3.10.12-1~22.04.5.
Enter the name for package 4:
libpython3.10-stdlib
Enter the version for package 4 needed for remediation:
3.10.12-1~22.04.4
The installed version is okay since it is version 3.10.12-1~22.04.5.
```

3. Kerberos
False Positive!

```
spy3@spyserver1:~/Scripts$ sudo ./vuln_remediation.sh
Enter the number of packages you want to search for:
4
Enter the name for package 1:
libgssapi-krb5-2
Enter the version for package 1 needed for remediation:
1.19.2-2ubuntu0.4
The installed version is okay since it is version 1.19.2-2ubuntu0.4.
Enter the name for package 2:
libk5crypto3
Enter the version for package 2 needed for remediation:
1.19.2-2ubuntu0.4
The installed version is okay since it is version 1.19.2-2ubuntu0.4.
Enter the name for package 3:
libkrb5-3
Enter the version for package 3 needed for remediation:
1.19.2-2ubuntu0.4
The installed version is okay since it is version 1.19.2-2ubuntu0.4.
Enter the name for package 4:
libkrb5support0
Enter the version for package 4 needed for remediation:
1.19.2-2ubuntu0.4
The installed version is okay since it is version 1.19.2-2ubuntu0.4.
```

4. Wget
False Positive!

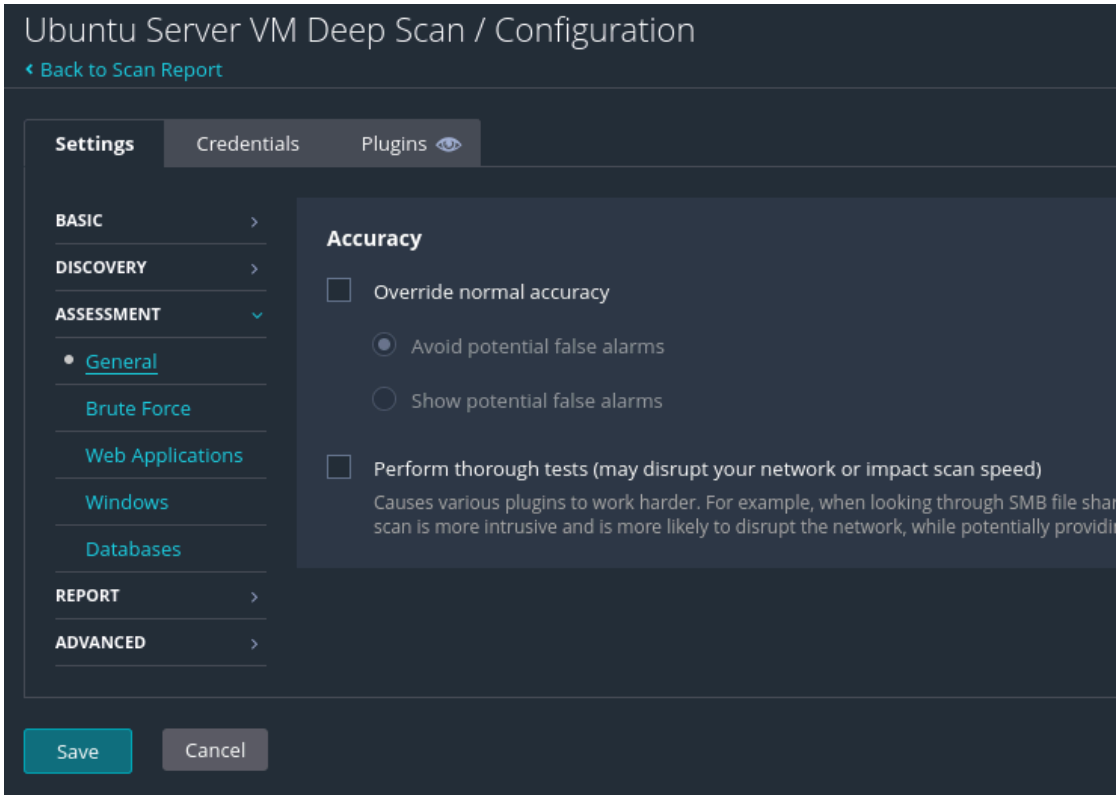
```
spy3@spyserver1:~/Scripts$ sudo ./vuln_remediation.sh
Enter the number of packages you want to search for:
1
Enter the name for package 1:
wget
Enter the version for package 1 needed for remediation:
1.21.2-2ubuntu1.1
The installed version is okay since it is version 1.21.2-2ubuntu1.1.
```

5. OpenSSl
False Positive!

```
spy3@spyserver1:~/Scripts$ sudo ./vuln_remediation.sh
Enter the number of packages you want to search for:
2
Enter the name for package 1:
libssl3
Enter the version for package 1 needed for remediation:
3.0.2-0ubuntu1.17
The installed version is okay since it is version 3.0.2-0ubuntu1.18.
Enter the name for package 2:
openssl
Enter the version for package 2 needed for remediation:
3.0.2-0ubuntu1.17
The installed version is okay since it is version 3.0.2-0ubuntu1.18.
```

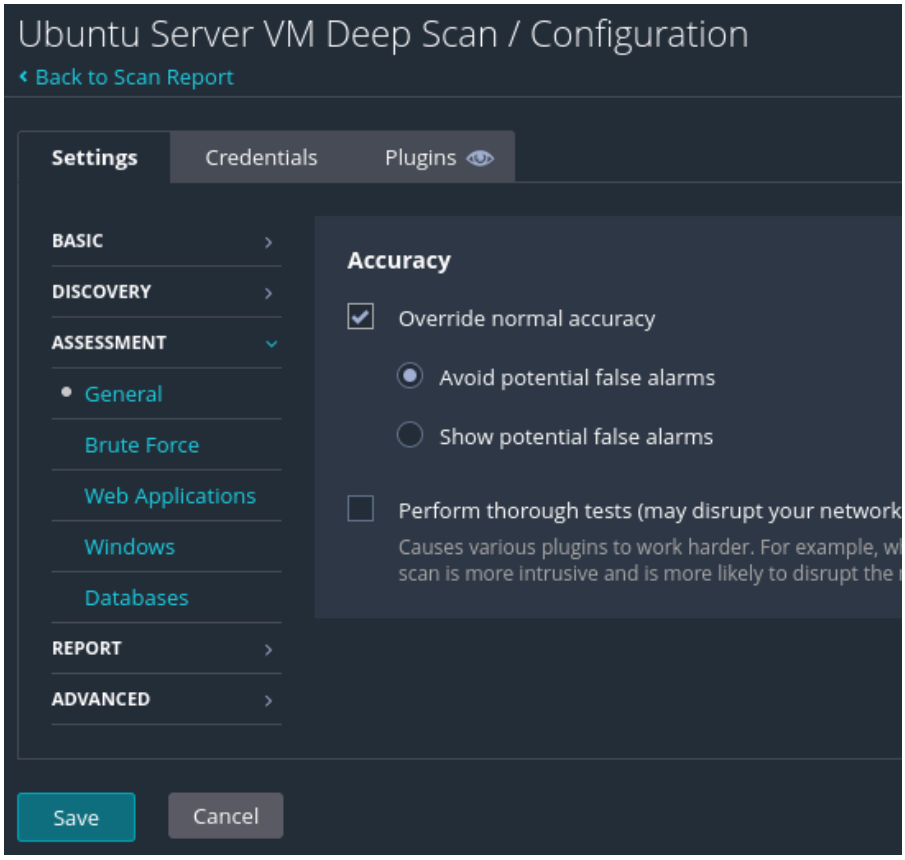
Why are all my critical vulnerabilities false positives?

I used a basic network scan and did not configure it whatsoever, I just used the default options. Apparently in order to avoid false positives you need to configure custom settings on the scan template!

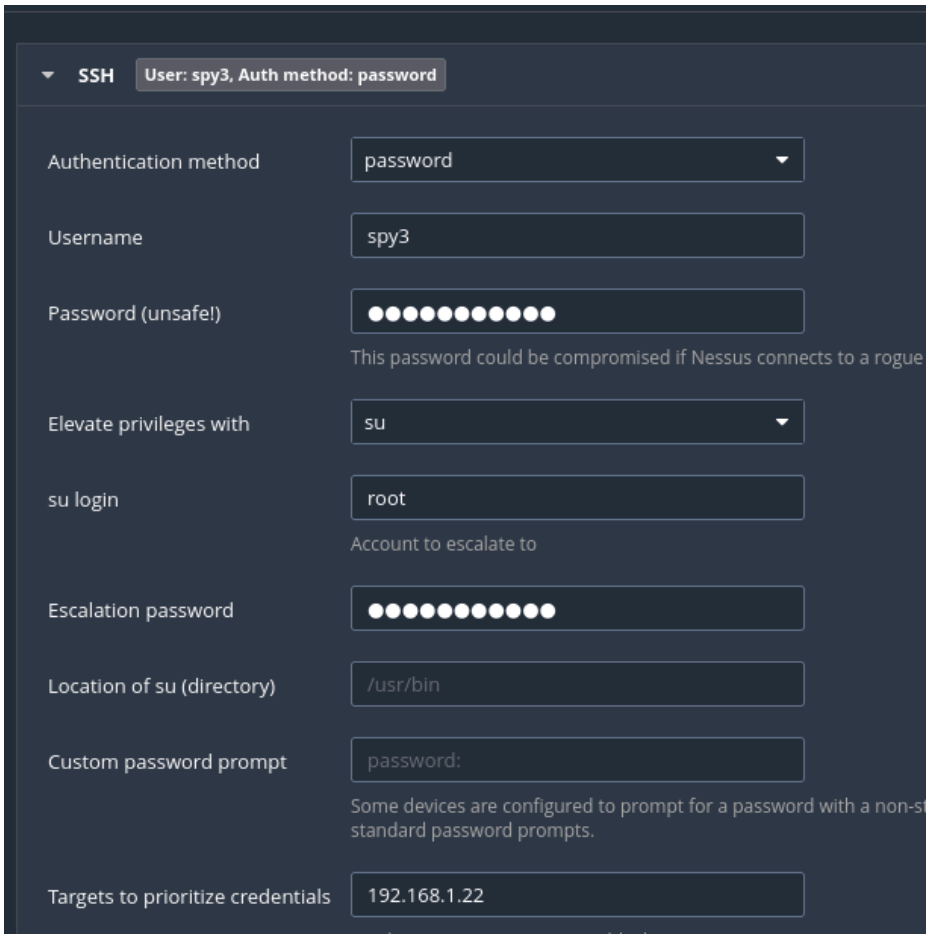


2. Scan the VM again to verify the vulnerabilities are gone:

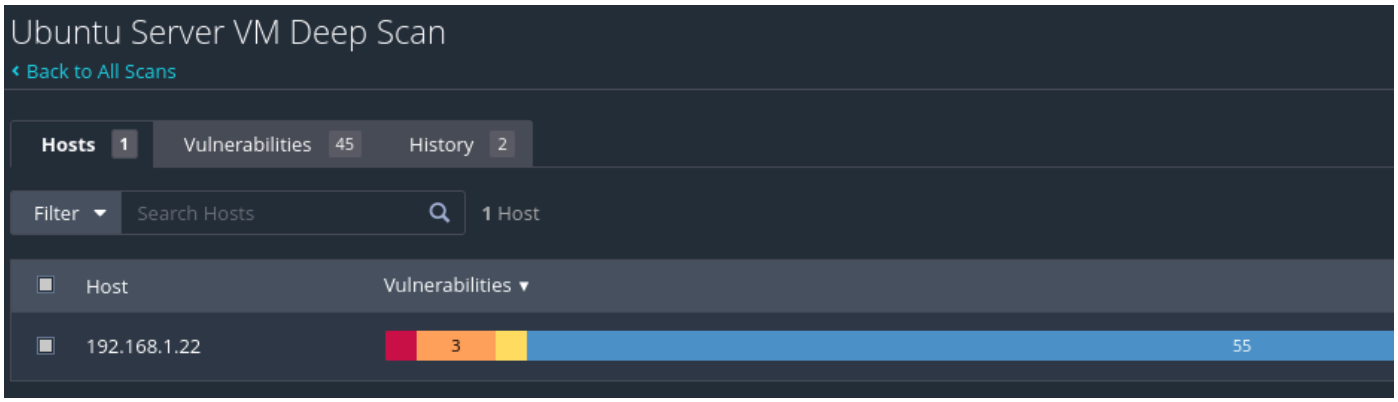
Since the critical vulnerabilities are possibly false positives, I will rescan the Ubuntu Server but I will tweak the scan settings so that false positives can be avoided.



The scan will also be credentialed again as well.



I will now run the new scan to show that the critical vulnerabilities were simply false positives:
The new scan has no critical vulnerabilities anymore! Only one high, a few medium and one low.



Well, I learned something very important from this. You should never just assume that the vulnerabilities are actually present and you should also try to configure scans to reduce false positives so that your time can be spent on real vulnerabilities.

Section #8 Reporting and Documentation:

The last step in the vulnerability management process is to perform reporting and documentation. During reporting and documentation you should explain what steps you took during the round of vulnerability management, what you discovered, and what you should do next time to improve the efficiency and effectiveness of the vulnerability management process. For this lab I will just discuss some things that I would need to do during the next scan to improve the process.

Improvements for my next scanning process:

1. I should use a custom basic scan template and configure the scan to limit the number of false positives it finds.

- During this round many false positives were detected on the Ubuntu Server VM and RHEL VM.
- A lot of time was spent tracking down false positives that could have been spent on remediating actual vulnerabilities.

2. Export data from Nessus to Excel for quicker analysis/ drilldown.
- I could have used Excel to analyze the vulnerability data more efficiently.

3. I can use CVSS 4.0 scoring for the vulnerabilities next time. I used CVSS 3.0 scoring for this lab but CVSS 3.0 is deprecated.