INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Reproducible modelling and simulating security vulnerability scanners evaluation framework towards risk management assessment of small and medium enterprises business networks

Ilias Chalvatzis[1], **Dimitrios A Karras**[2]*, **Rallis C Papademetriou**[1]

1 University of Portsmouth, UK, School of Energy & Electronic Engineering, Anglesea Building, Anglesea Road, Portsmouth (UK), PO1 3DJ
2 School of Science, General Department, National and Kapodistrian University of Athens, Athens

OPEN ACCESS

Received: 09-07-2020
Accepted: 01-08-2020
Published: 16.10.2020

Editor: Dr. Natarajan Gajendran

Citation: Chalvatzis I, Karras DA, Papademetriou RC (2020) Reproducible modelling and simulating security vulnerability scanners evaluation framework towards risk management assessment of small and medium enterprises business networks. Indian Journal of Science and Technology 13(37): 3910-3943. https://doi.org/10.17485/IJST/v13i37.868

* **Corresponding author**.

dimitrios.karras@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: Risk Management has been recognized as a critical issue in computer infrastructures, especially in medium to large scale organizations and enterprises. The goal of this research report is to provide a practical comprehensive virtual machine based framework for assessing the performance of vulnerability scanners applied to such enterprises, focused to small and medium size ones towards a risk evaluation analysis. Moreover, the purpose of this paper is to compare three of the most well-known free vulnerability scanners (Nessus, OpenVAS, Nmap Scripting Engine) with regards to how they can be used to systematise the process of Risk Assessment in an enterprise, based on the herein presented experimental evaluation framework involving virtual machine testing. **Method:** The proposed methodology is based on developing a framework for suitable setup and usage of virtual machines making risk analysis practical and being capable of comparing different vulnerability scanners. **Findings:** The herein developed framework is shown to be efficient with regards to comparison and selection of candidate risk analysis software with easily accessed and affordable infrastructure. **Novelty:** Although there might be few other similar comparisons of vulnerability scanners in the literature, the main herein contribution is the provision of a practical and above all easily reproducible framework for small business enterprises to establish proper selection procedures of such security software without spending a lot of money for expensive testing infrastructure.

Keywords: Vulnerability Scanning; risk assessment; nessus; OpenVAS; Nmap scripting engine

1 Introduction and Related Work

A vulnerability scanner is a software application that evaluates security vulnerabilities in networks or host systems and reports a set of scan results. Those vulnerabilities can be software bugs and backdoors, missing OS patches, insecure configurations and vulnerable ports and services [1].

It allows early detection of known security problems but by itself it isn't the perfect solution for the protection of the network because it can only provide a snapshot of the state of the security of a network when each scan is concluded. Furthermore, the scan results can be overwhelming on large networks and human judgment is needed while reading the results to separate true vulnerabilities from false positives. Finally, vulnerability scanners operate by using plugins that discover known vulnerabilities only which can lead to false negatives due to missing plugins or 0-day exploits. [1,2]

The data that vulnerability scanners produce, especially on bigger networks can be overwhelming and it can be difficult to determine the overall security state of a network or a cluster of it that is deemed more important. A way to aggregate the results of vulnerability scanning to produce an overall report for the security of the system would be very useful to the security specialists to monitor the state of the network and where, as well as the priority of actions that need to be taken. [1,2]

While this would be a very good first step to help the security specialists in allocating time and resources more efficiently in securing the network a further step could be taken to help them be one step ahead of the attackers, taking into account temporal aspects and dynamic properties of vulnerabilities to create a stochastic model that can "predict the future" by anticipating security gaps on the network that an attacker would aim to exploit and optimizing resource allocation to ensure the efficient protection of key business assets. [1,2]

The aim of this report is first to provide a comprehensive framework for the vulnerability analysis assessment of small to medium size enterprises and, secondly, to analyze the free and open available vulnerability scanners in order to adopt the ideal ones as the base of creating such a stochastic model and based on its features to identify how this could potentially be elaborated. Although, there are other attempts in the literature [3–8] to provide assessments of vulnerability scanners based on their features and characteristics, offering relevant comparison frameworks, the contribution of this paper lies in the development of a suitable comprehensive, practical and above all reproducible evaluation framework towards modelling and emulating enterprise environment risk assessment. It is based on virtual machine vulnerability performance analysing and focuses on modelling and simulating the business environment of a small to medium size enterprise, extending significantly the scope and results of the authors preliminary study in [1], contributing the complete setup and framework risk assessment development and analysis.

2 The Vulnerability Scanners and Proposed Solutions

In general, the Vulnerability scanners consist of four main modules: the user interface, the scan engine, the vulnerability database and a report module.

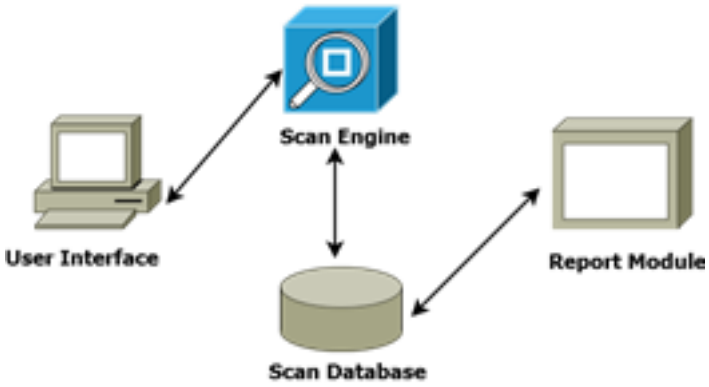


Fig 1. Vulnerability scanner architecture

There are lots of Vulnerability scanners, some of them are free and Open Source (or have a free version with limitations) while others can be very expensive. For our purposes, the main requirement is being able to create custom tests and custom reports by combing the results with our own metrics and formulas to produce a summary of the total security of the network. Alternatively, the reports should be able to be exported in a way that the data is retrievable by an external program if we want to perform the generation of the overall security mark and the proposed stochastic model externally.

In Figure 2 we present the two proposed solutions on how a Vulnerability scanner could be used to produce an overall security mark for the network. The blue colored shapes are unmodified modules of the vulnerability scanner while the gray shapes are where we would have to create custom code.

Fig 2. Using a vulnerability scanner to produce an overall security mark

In order to aggregate the data produced by the vulnerability scanner we have two options.

The first option (Option A) is to do the aggregation of the results and the production of a final report within the scanner environment which would require the scanner to give us the ability to produce custom reports with scripting that allows the execution of functions that use the scan data and produce custom text.

Option B would take the raw data from the Vulnerability Scanner's default report and import it into another software that would then process it to produce our final assessment of the overall security of the network. This would require the data of the report produced by the scanner to be structured in a way that's easily exportable and comprehensible by other software.

Ideally the chosen Vulnerability Scanner would provide both of these options. For this reason, the Vulnerability scanners we'll be examining are going to be mainly Open Source and free with the exception of Nessus that used to be Open Source and is still one of the most widely used and well documented Vulnerability scanners.

Nessus

Nessus has been one of the most popular Vulnerability Scanners mainly due to the fact that it initially was Open Source and Free up until 2005 where they closed the source code in 2005 and removed the free version in 2008. There is a free "Nessus Home" version but it is only available for personal use in a home environment and it has a few limitations like only allowing the scanning of 16 IPs per scanner. It has an extensive vulnerability database that is updated on a daily basis with over 90.000 different plugins at this point and is available for Windows (Server and 7, 8, 10), various Linux Distributions and FreeBSD.

Nessus will be the standard that we compare the other vulnerability scanners. Most of its features (client/host architecture, smart service recognition, and CVE compatibility) should be expected from the other vulnerability scanner solutions too and its vulnerability/plugin database numbers is a good sample of what we should be looking for on a good vulnerability scanner. The most important aspect though that makes Nessus a good solution and the standard of comparison is the existence of the Nessus Attack Scripting Language. We will look further in NASL and its custom scripts in another section and any other scanner that could be considered should provide a similar ability to create custom tests and reports.

OpenVAS

OpenVAS is the free, open source Vulnerability scanner that forked from the GPL version of Nessus (Version 2) after it went proprietary in 2005. What is important for our purposes is that OpenVAS plugins are still written in Nessus Attack Scripting Language (NASL). The actual security scanner is accompanied with a regularly updated feed of Network Vulnerability Tests (NVTs), over 50,000 in total. All OpenVAS products are Free Software. Most components are licensed under the GNU General Public License (GNU GPL)."

It being open source and free while using NASL makes it a great alternative for our purposes.

Nmap Scripting Engine (NSE)

Nmap (Network Mapper) is a free and open source utility for network discovery and security auditing that is considered one of the most popular security tools due to its flexibility, power, portability and ease of use. It runs on all major computer

Operating Systems and official binary packages are available for Linux, Windows and Mac OS X. The Nmap Scripting Engine (NSE) allows users to write their own scripts to automate a wide variety of networking tasks through Nmap.

Even though NSE isn't a comprehensive vulnerability scanner (the number of available scripts for NSE is nowhere near those for Nessus and OpenVAS with about 600 scripts available at the time of writing this), the vulnerability exploitation capabilities of NSE, its integration with the rest of the libraries and tools of the Nmap suite, and the simple interface that the "vuln" NSE library provides to create well-organized vulnerability reports that can also contain CVE references can be proven very useful for our purposes.

3 Virtual Machine Testing Setup and Details

In order to test the scanners a virtual machine network was created. While Nessus and NSE can be run in a Windows environment, OpenVAS server requires a Unix Operating system. For this reason the initial testing environment consisted of four virtual machines to represent a small business network (a Windows 10 workstation, a windows 7 workstation, an Windows 2012 server and an Ubuntu server) as well as two vulnerability scanning machines, one for the Nessus/NSE server (Windows 7) and the other for the OpenVAS server (Kali Linux) For this particular comparison the free Oracle VirtualBox Environment was used to create these 6 machines with a NAT network setup so they can connect to each other as well as the internet.

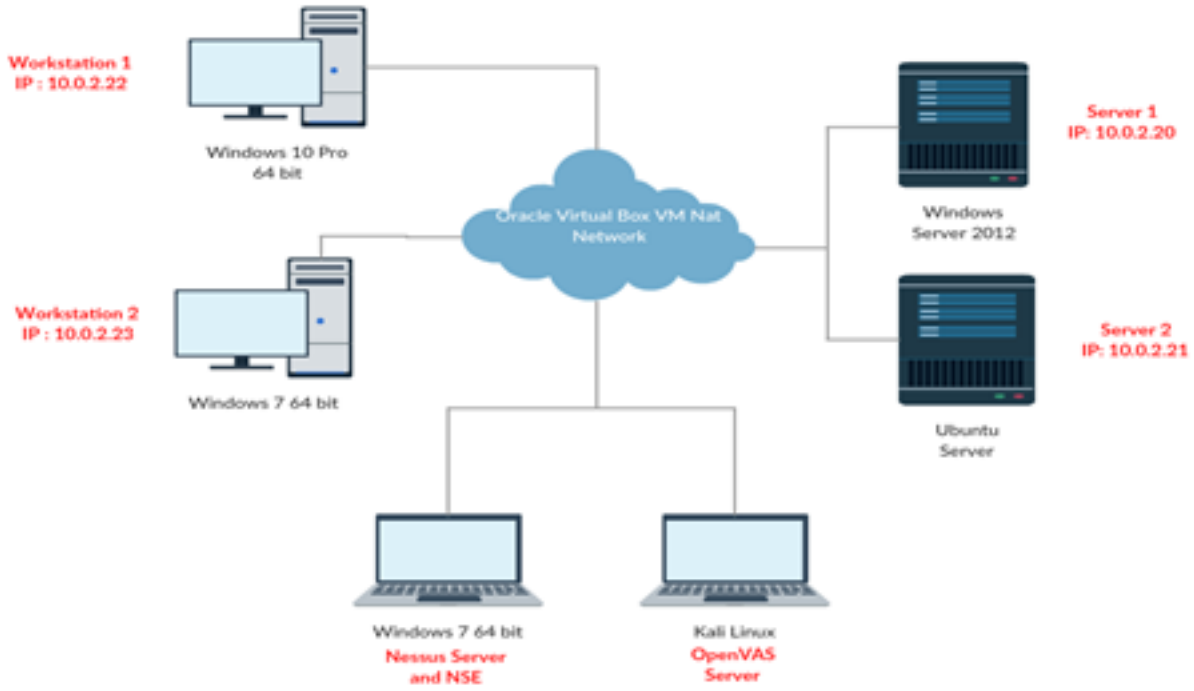


Fig 3. Virtual machines setup for testing the vulnerability scanners

Overall, the easiest vulnerability scanner to setup, use and add custom scripts was NSE by installing the Zenmap GUI in a Windows environment along with Nmap. The Nessus server was easier to setup (as there is a windows installation) but the fact that both scanners can be run from a web interface from any kind of OS as long as it can connect to the server makes this a non-issue on a business environment.

By performing the vulnerability scanning with each scanner on the test network the results showed that the scanners are not that far in comparison with each one having different pros and cons.

NSE was the fastest by completing the scan of the whole network in 10 minutes while Nessus took twice this time and OpenVAS took 1 hour and 20 minutes. Nessus found the most vulnerabilities by finding three Critical vulnerabilities when OpenVAS and NSE both found the same two. Presentation and options wise Nessus and OpenVAS are similar with Nessus having a slight edge while NSE is more barebones than the other two in those aspects.

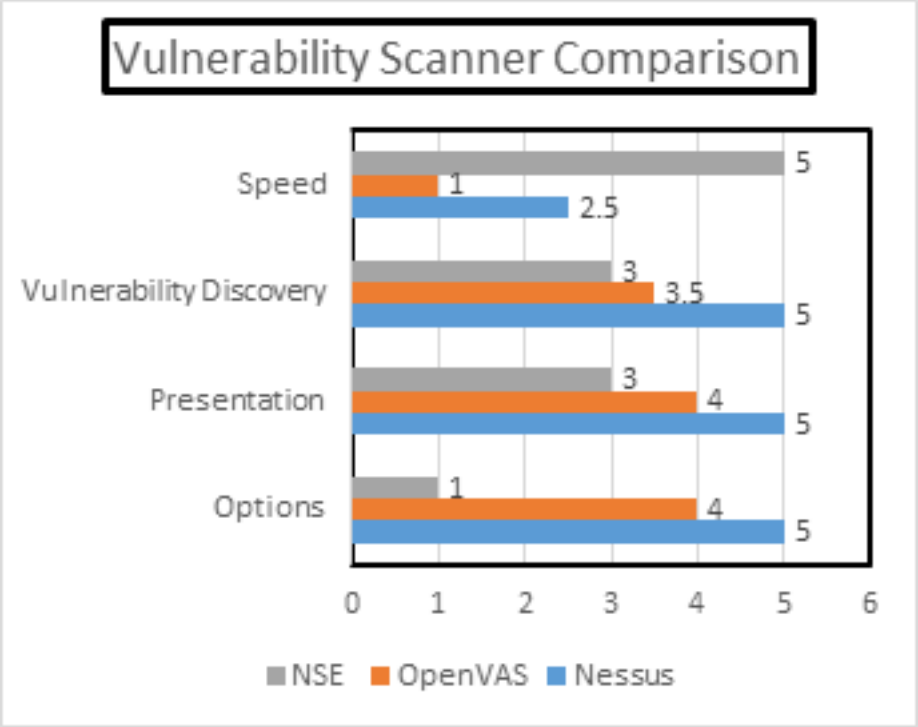


Fig 4. Vulnerability scanners comparison

The machines have to be created in the Virtual Box environment and in order for them to communication with each other a NAT Network can be set up.

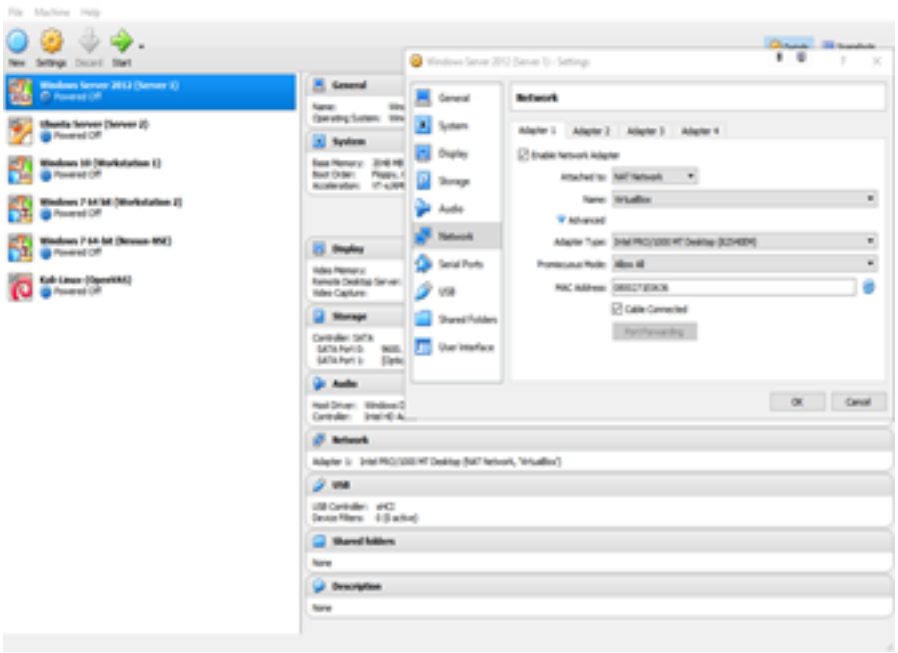


Fig 5. Virtual box creation of the virtual machines

Server roles in windows server 2012 (the services it will run) can be enabled by a simple menu

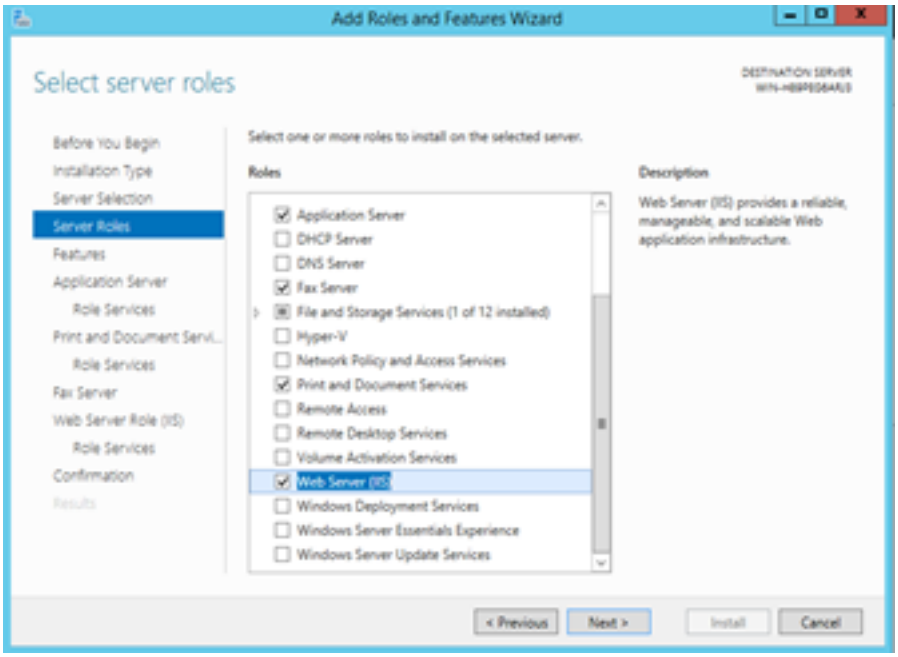


Fig 6. Windows server 2012 selection of roles

Figure 6 shows the windows server 2012 running the selected roles

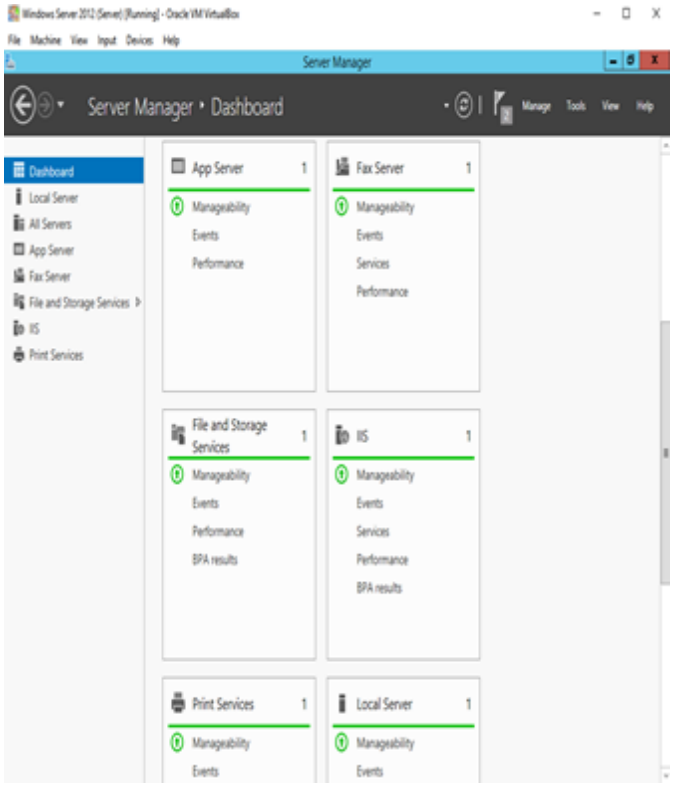


Fig 7. Ubuntu server selection of roles

The roles of the Ubuntu Server can be selected during the installation process as shown in Figure 7.

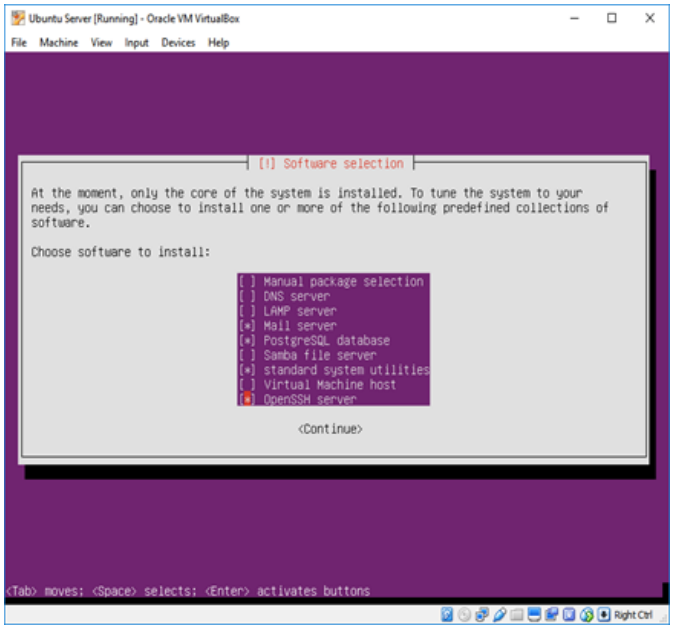


Fig 8. Ubuntu server services

Figure 8 shows the services the Ubuntu server is running

In order to keep the test results consistent we should set up static IPs for our machines. Figure 9 shows how this is done on Windows.

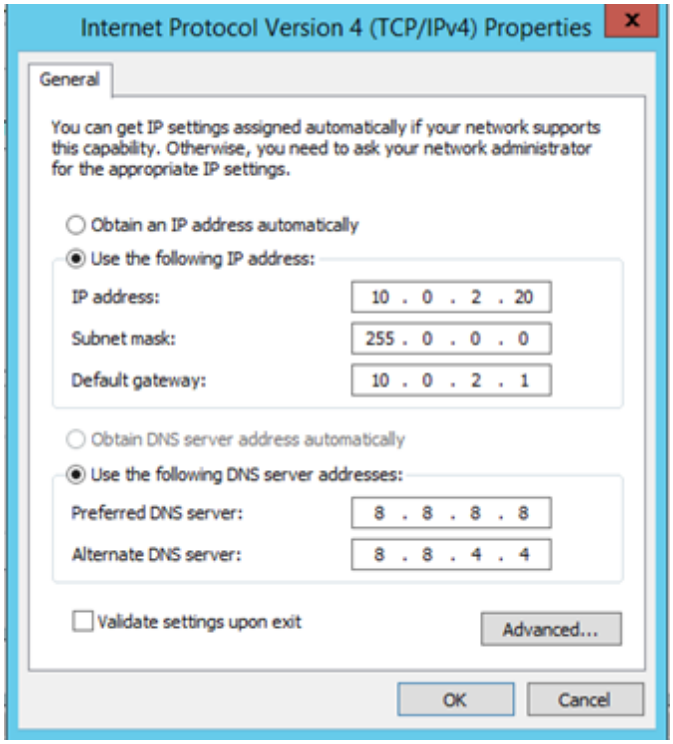


Fig 9. Static IPs set up for the virtual machines on Windows

The process is slightly more complicated on the Ubuntu server having to modify the interfaces file as in Figure 10

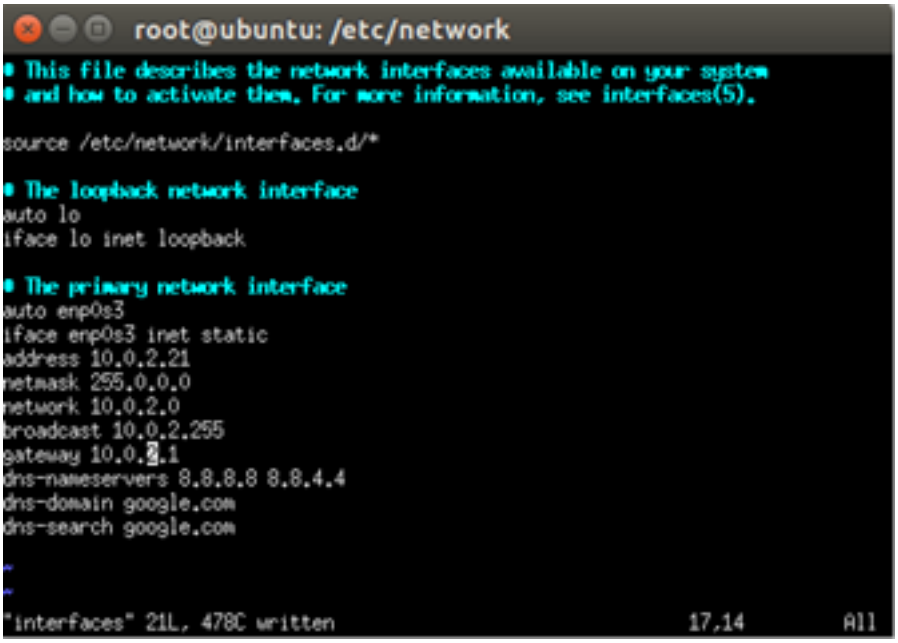


Fig 10. Static IPs set up for the virtual machines on Ubuntu

The Nessus/NSE machine can communicate with the 4 machines of the network through their static IPs as shown below.

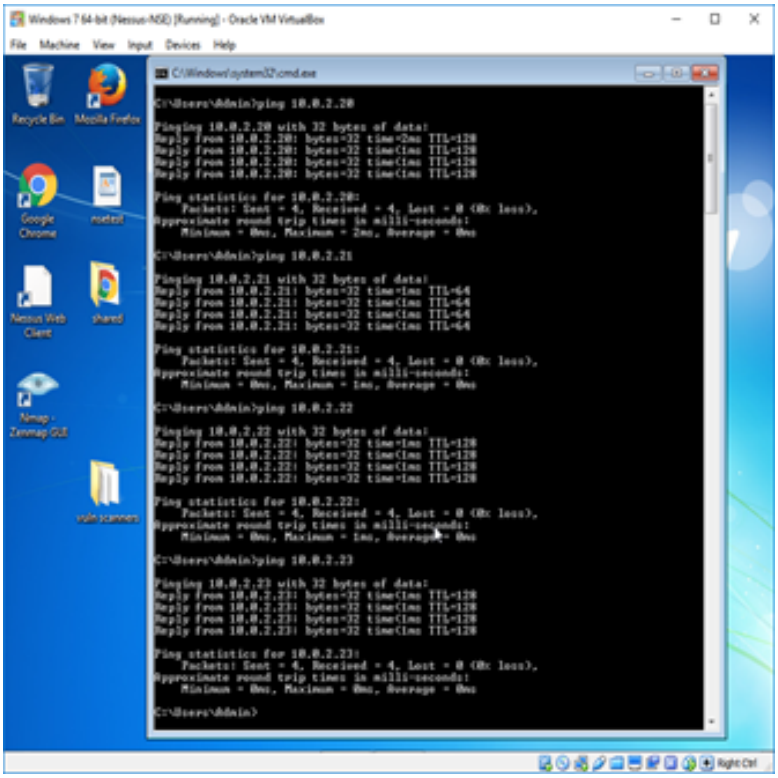


Fig 11. Nessus/NSE machine communication with the 4 machinesof the network through their static IPs in Windows

And, similarly, the Kali Linux machine can perform , where the OpenVAS server is running

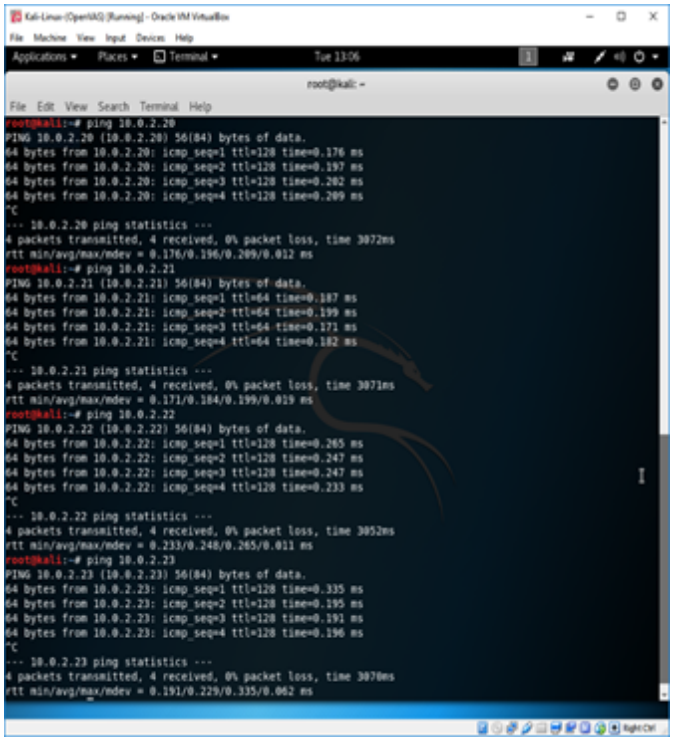


Fig 12. OpenVas machine communication with the 4 virtualmachines of the network through their static IPs in Kali Linux

Workstation 1 is a fresh install of Windows 10 downloaded from the Microsoft website so it has all current updates but no extra software so it shouldn't be expected to have serious problems.

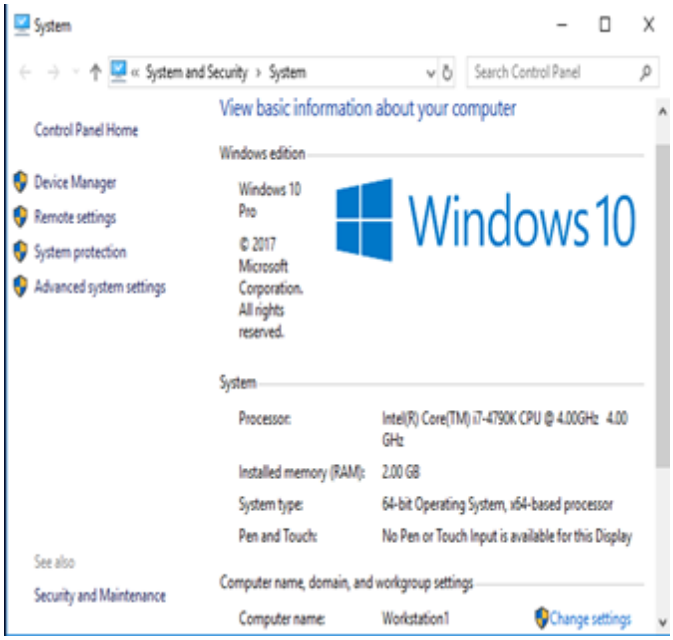


Fig 13. Workstation 1 configuration

Workstation 2 has Windows 7 Pro Service pack 1 installed an a lot of out of date software with network access so the vulnerability scanners should find vulnerabilities.

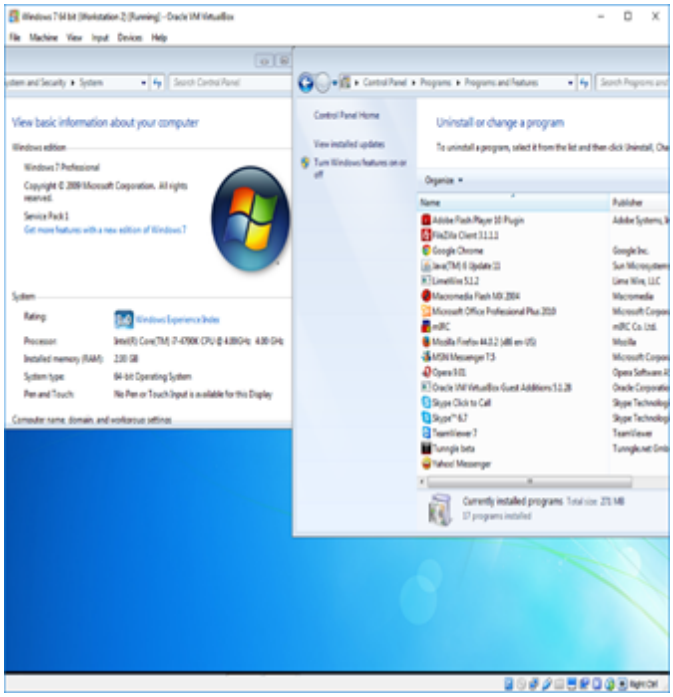


Fig 14. Workstation 2 configuration

3.1 Scanning the VM network with Nessus

The targets of the Nessus scanner can be specified as a range in our case 10.0.2.20-10.0.2.23

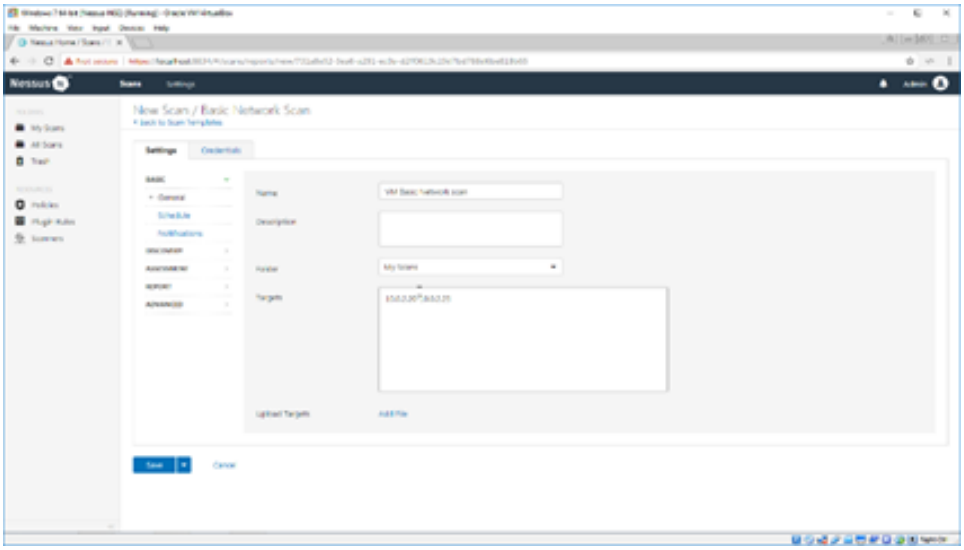


Fig 15. Targets of the Nessus scanner

Below are the settings for a basic network scan

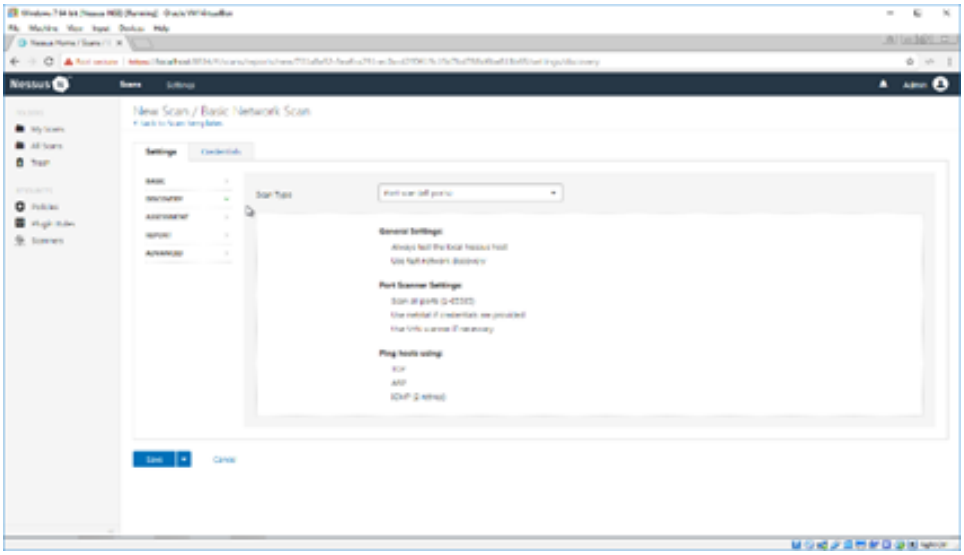


Fig 16. Basic network for the Nessus scanner set up

The scan of the whole network took Nessus 20 minutes and as expected more vulnerabilities where found in the Windows 7 Workstation and the least on the Windows 10 fresh install.

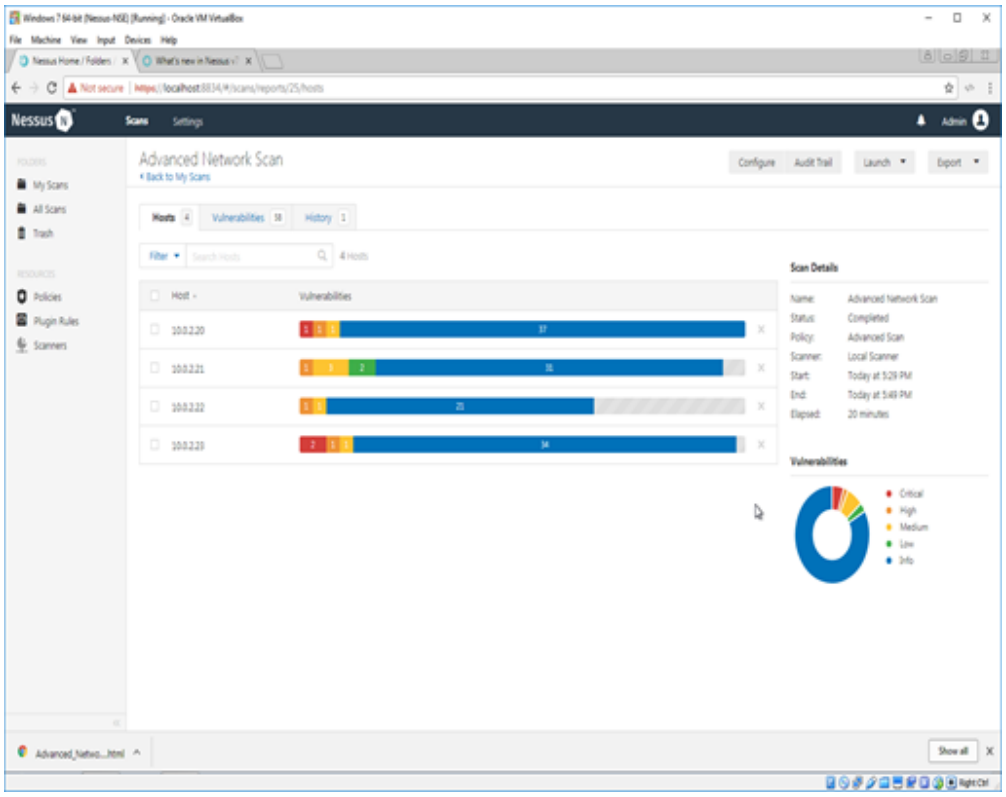


Fig 17. Vulnerabilities detection on Windows 2012 server

MS15-034 Vulnerability was the Critical Vulnerability found on the Windows 2012 server

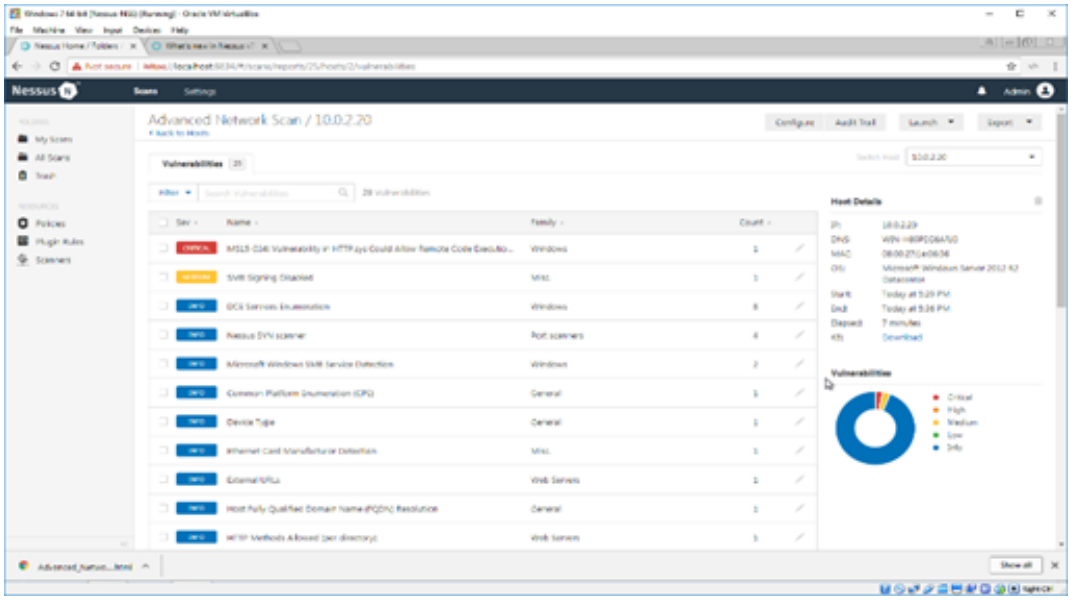


Fig 18. Critical vulnerabilities detection on Windows 2012 server (cont.)

This vulnerability can be fixed with a patch which wasn't installed on our fresh installation and has a CVSS Base score of 10 and Temporal Score of 7.8

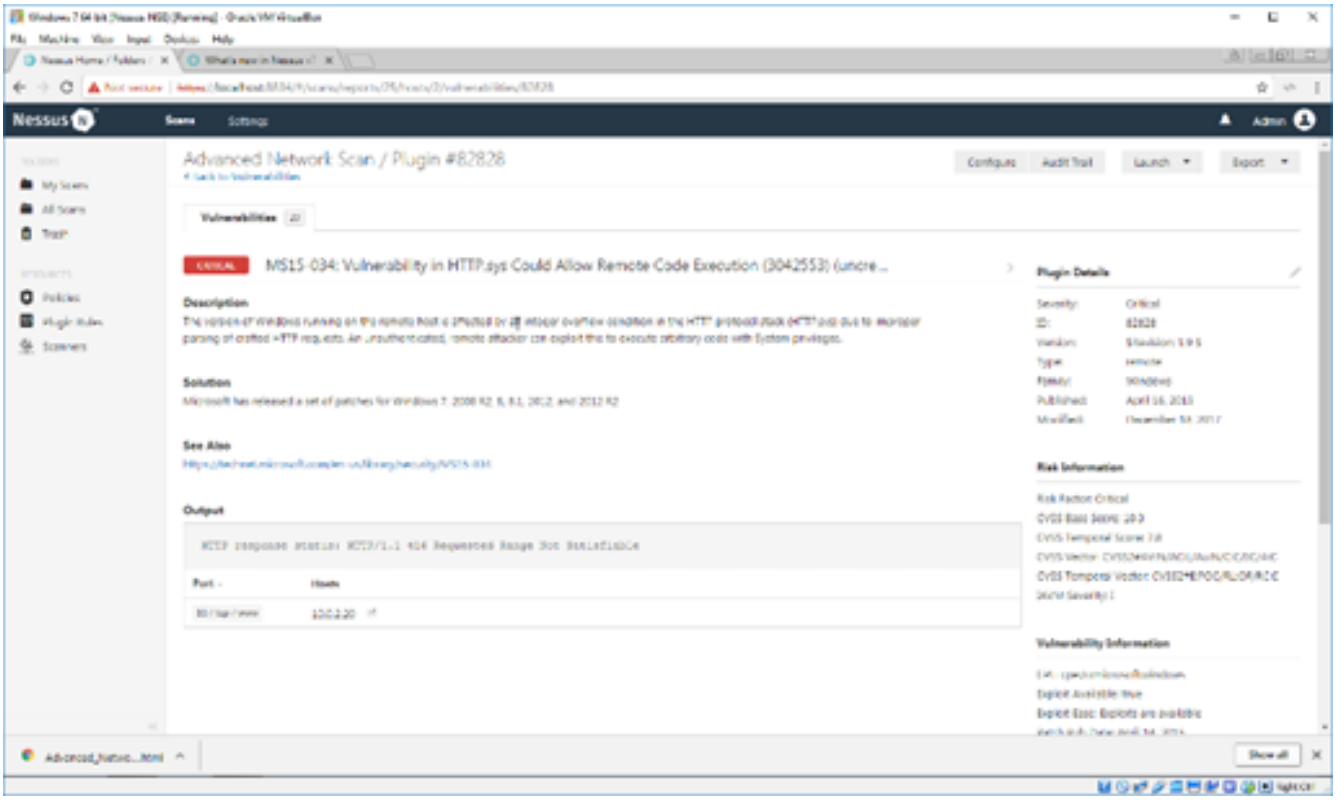


Fig 19. Vulnerabilities fixing

The Linux Server had mainly SSL certification vulnerabilities

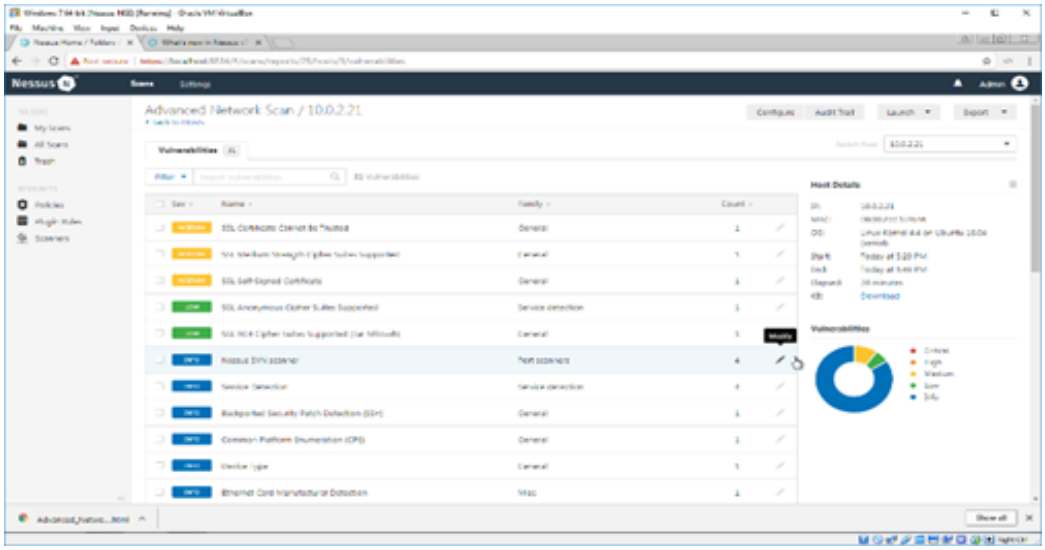


Fig 20. Vulnerabilities detection in Linux server

A proper SSL certificate was not created for our server and this creates a vulnerability with Medium Risk factor and CVSS Base score of 6.4

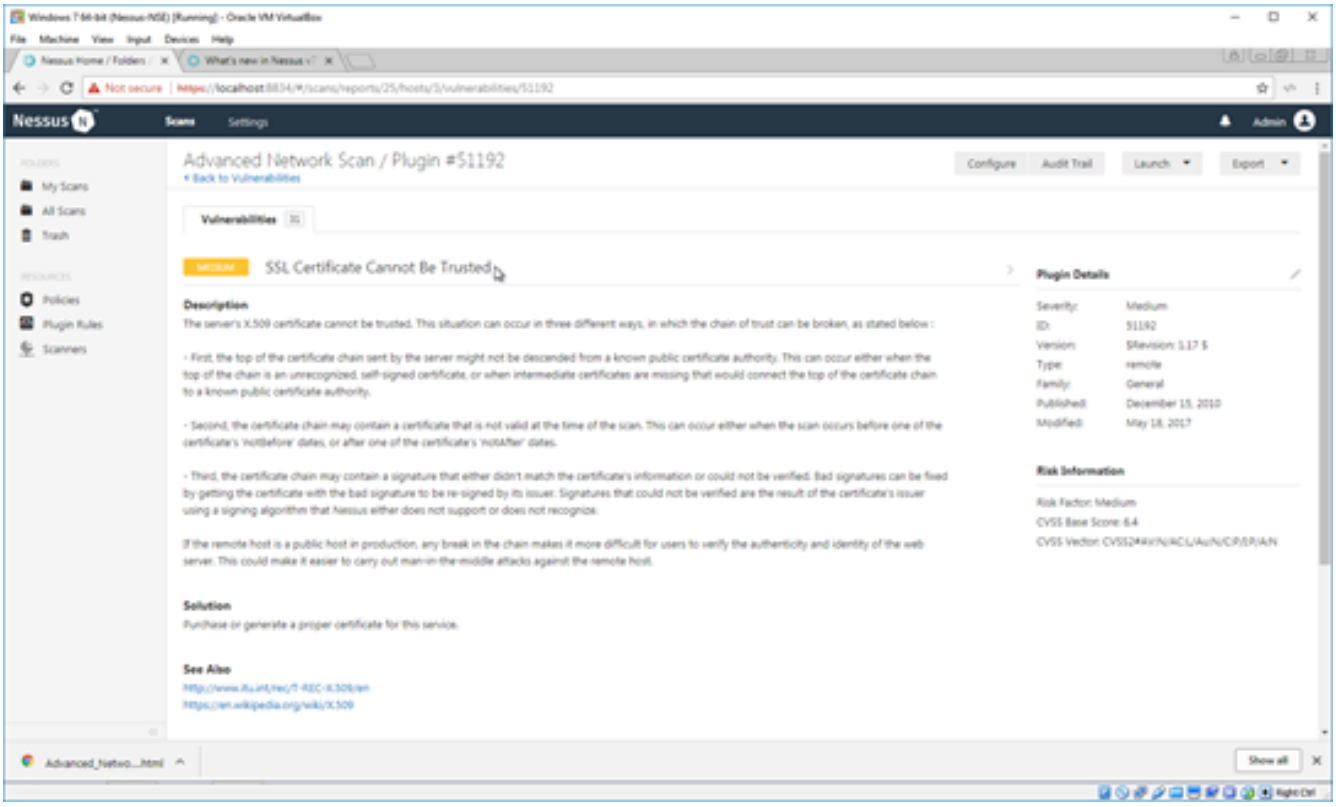


Fig 21. SSL certification vulnerabilities detection in Linux server

The scanner also reported on the services running on the Ubuntu server (SSH,SMTP, POP3 and IMAP as expected since we are running SSH and Mail servers on this machine)

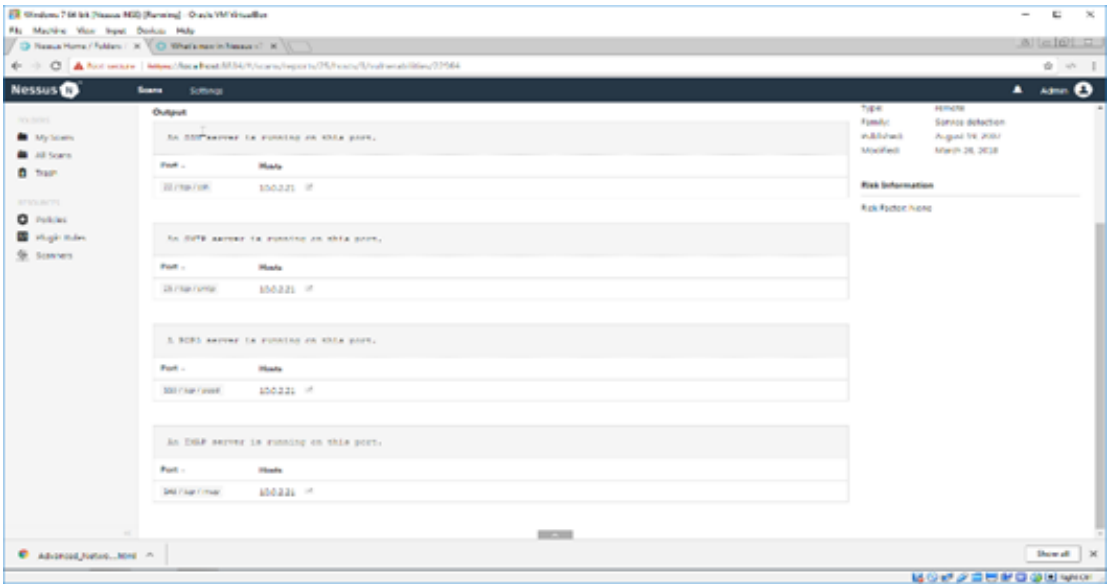


Fig 22. Services report on Linux server

OS Identification also worked as expected

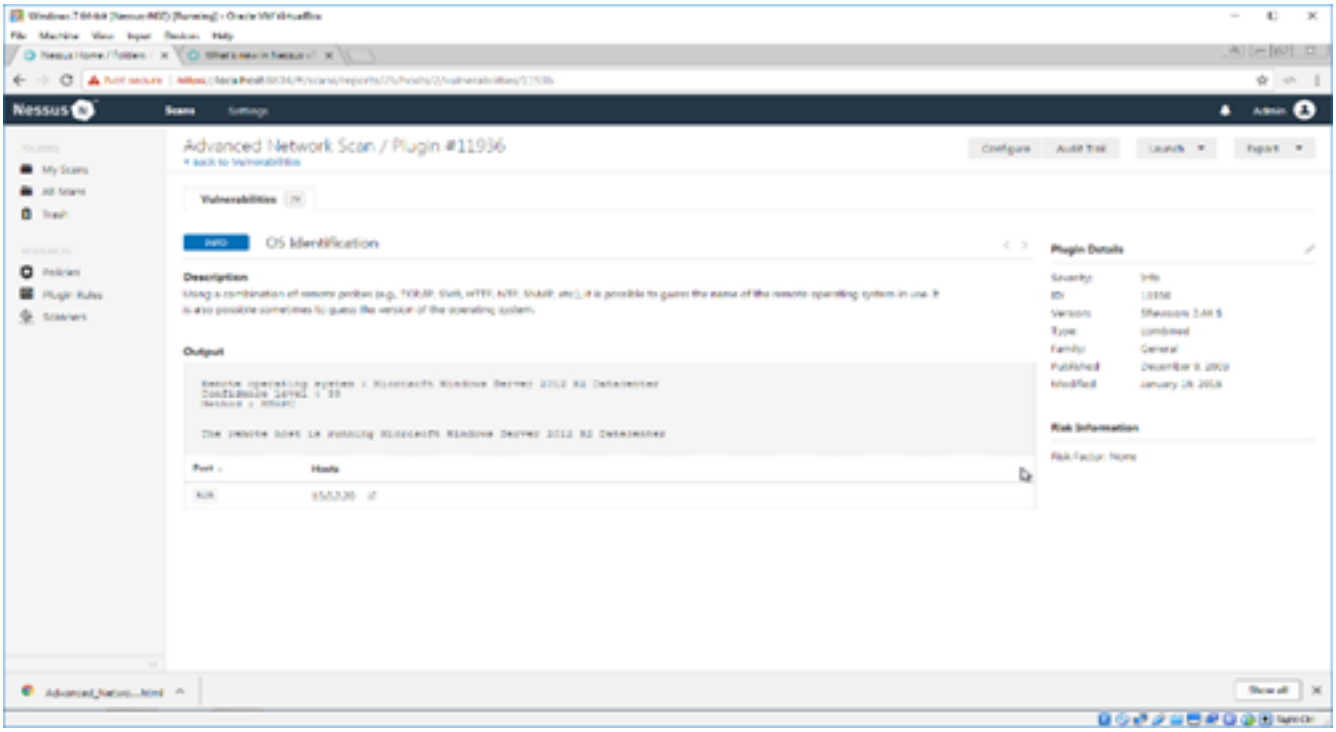


Fig 23. Workstation 1 vulnerabilities outcome

The Windows 10 fresh install didn't really have any vulnerabilities except from SMB Signing being disabled as expected from a fresh install

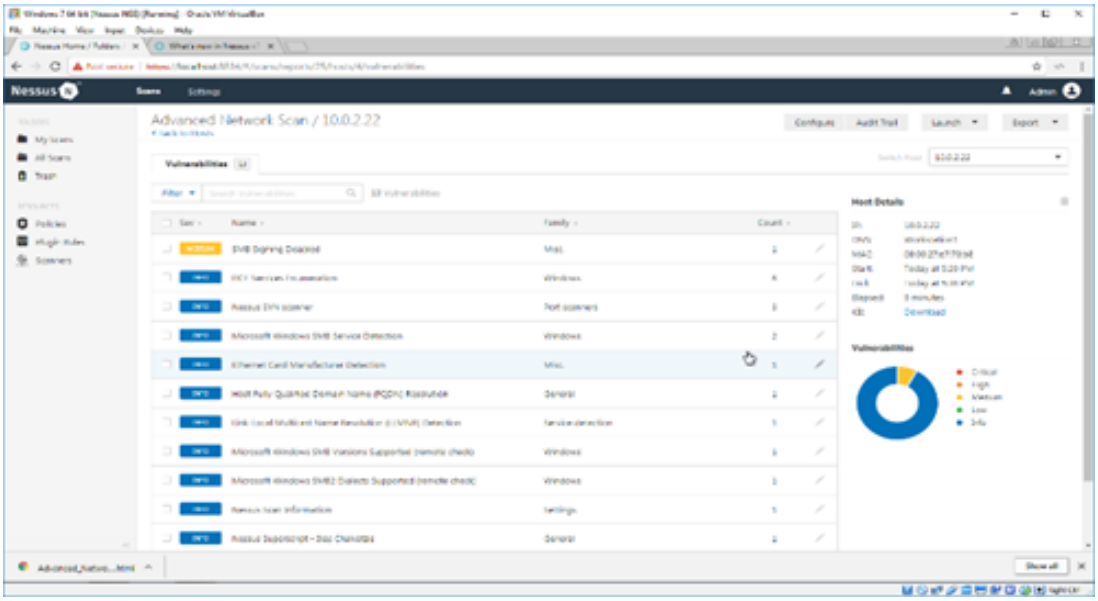


Fig 24. Workstation 2 vulnerabilities detection

The outdated Windows 7 machine on the other hand had 2 critical Vulnerabilities

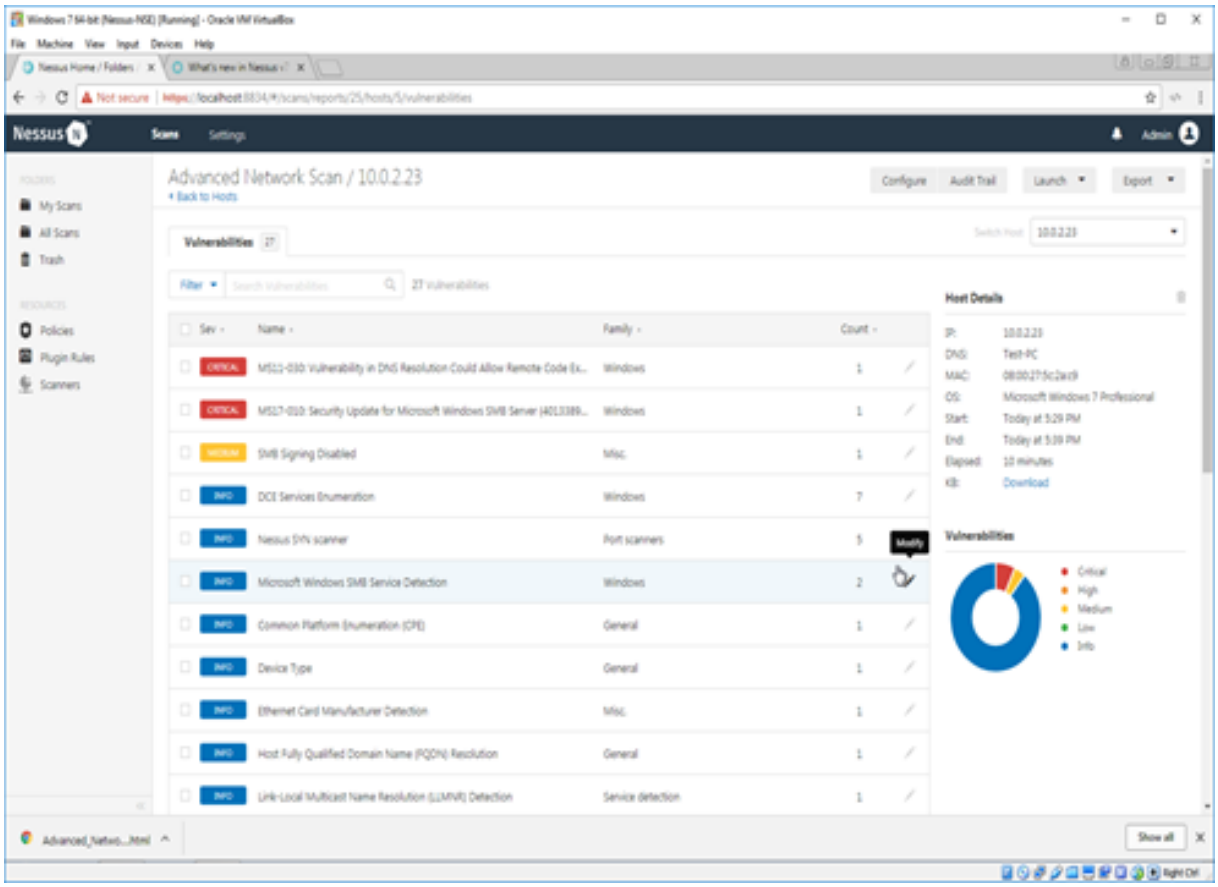


Fig 25. Workstation 2 critical vulnerabilities detection

The first one was MS11-030 a vulnerability in DNS resolution with a CVSS Base score of 10 and Temporal of 7.8 that was also fixed with a patch that we didn't install

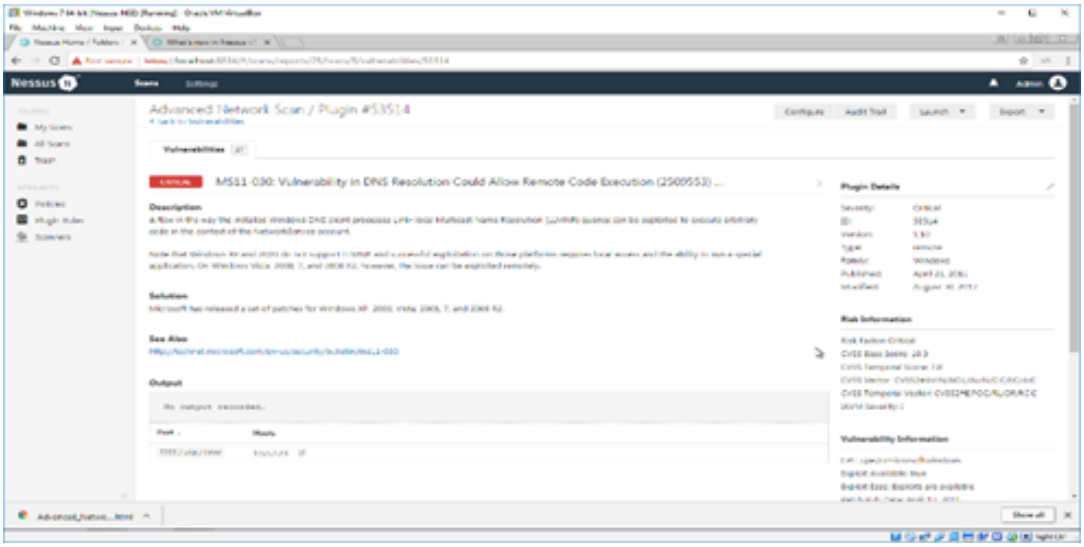


Fig 26. Workstation 2 critical vulnerabilities analysis 1

The second one was MS17-010 with a CVSS Base Score of 10 and temporal score of 9.5 and it makes the system vulnerable to the famous Wannacry ramsomware

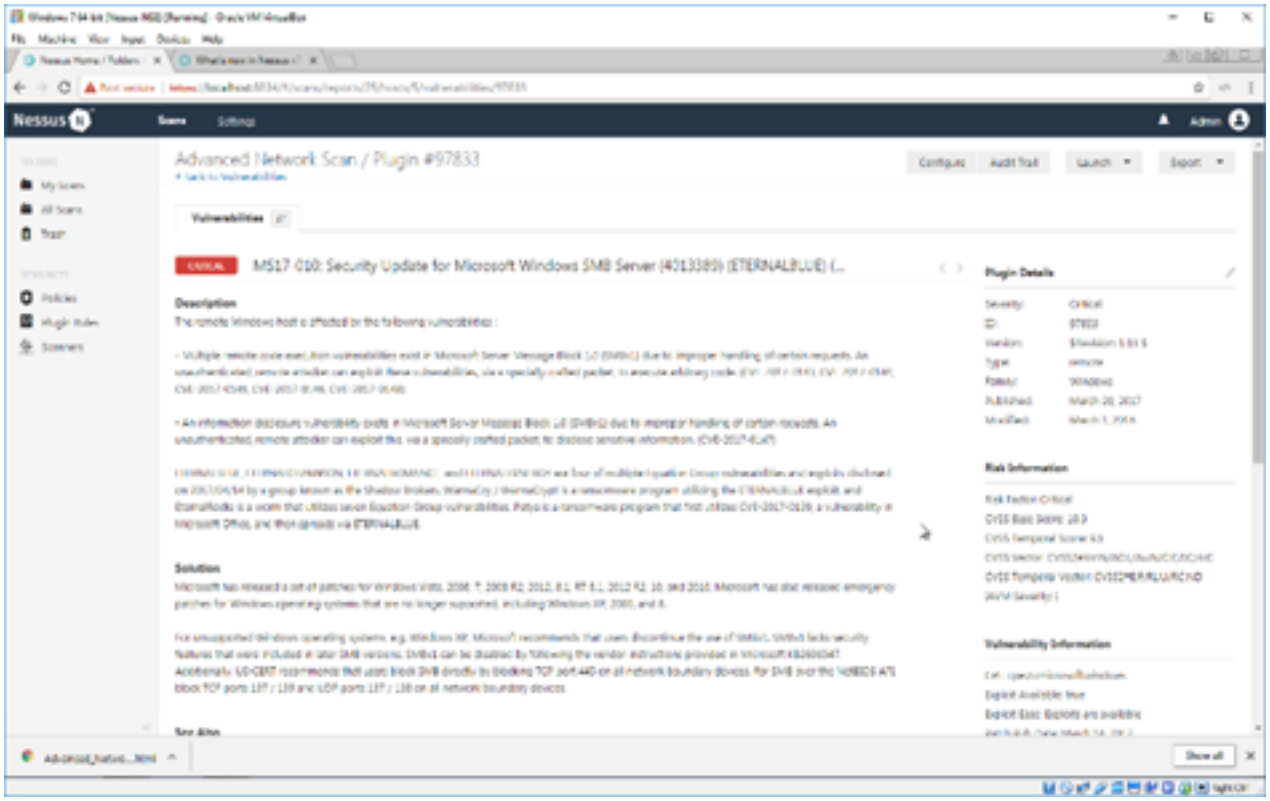


Fig 27. Workstation 2 critical vulnerabilities analysis 2

Below is the executive summary giving an overview of the network vulnerabilities

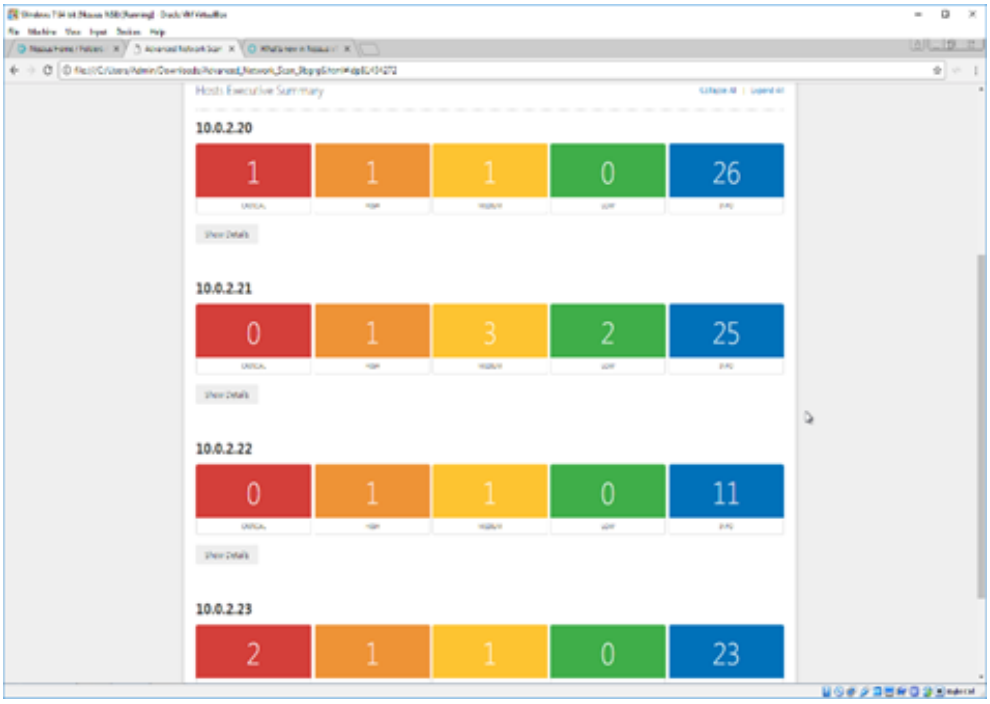


Fig 28. Executive summary, overview of the network vulnerabilities

This can be expanded to show more details on each system. The High vulnerability called "Nessus Superscript" exists in all systems and is just a test custom script so it can be ignored

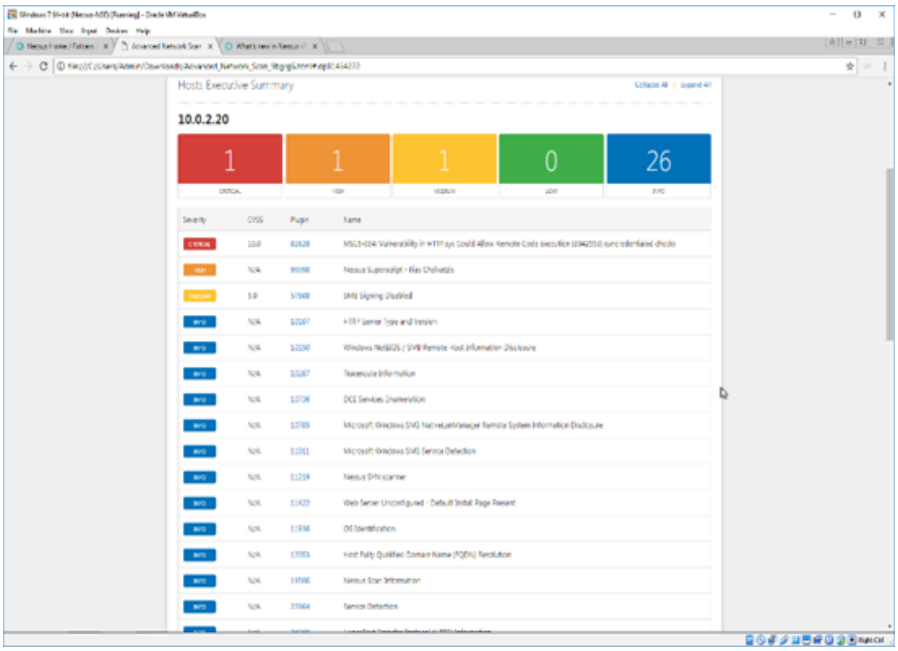


Fig 29. Executive summary, overview of the network vulnerabilities expansion to specific virtual machines

The vulnerabilities themselves can be expanded to show more details like CVSS scores and description / fixes for the vulnerability

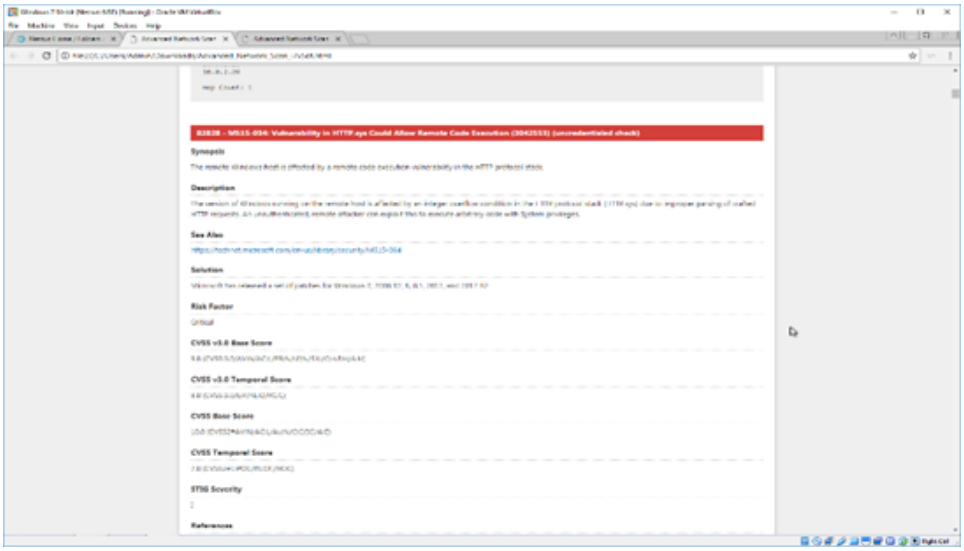


Fig 30. Executive summary, overview of the network vulnerabilities CVSS metrics

This report can be exported in html and CSV formats.

3.2 Scanning the VM network with OpenVAS

OpenVAS can also be set up to scan a range and has similar options with Nessus in regards to scanning. The scan in OpenVAS took 1 hour and 24 minutes to complete, that's over 4 times as much as Nessus.

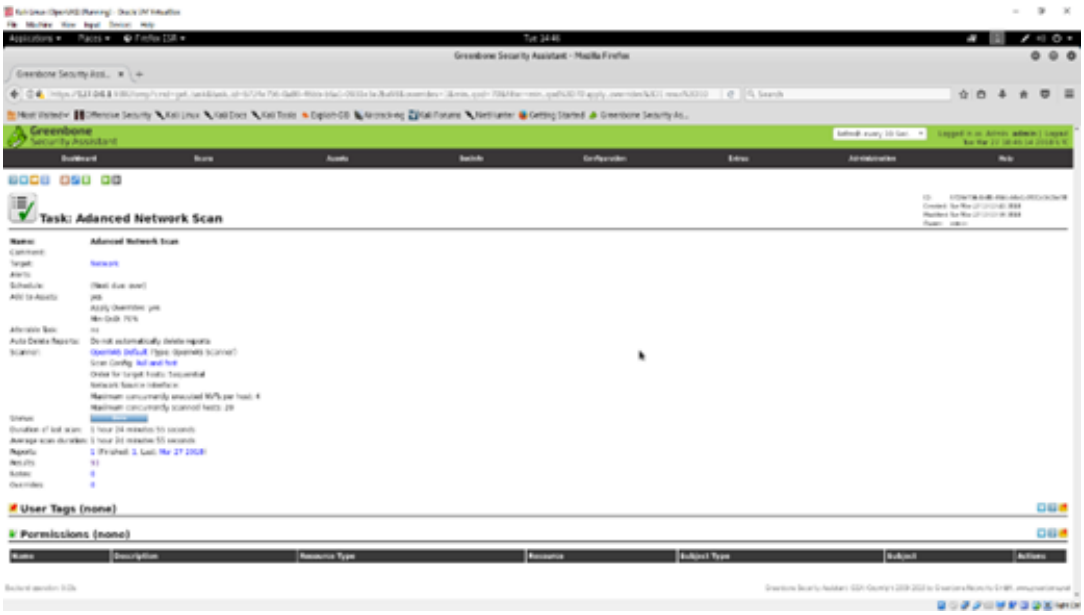


Fig 31. OpenVAS vulnerabilties detection

Two Critical Vulnerabilities were found in the Windows 2012 server and the others on the Windows 7 Workstation

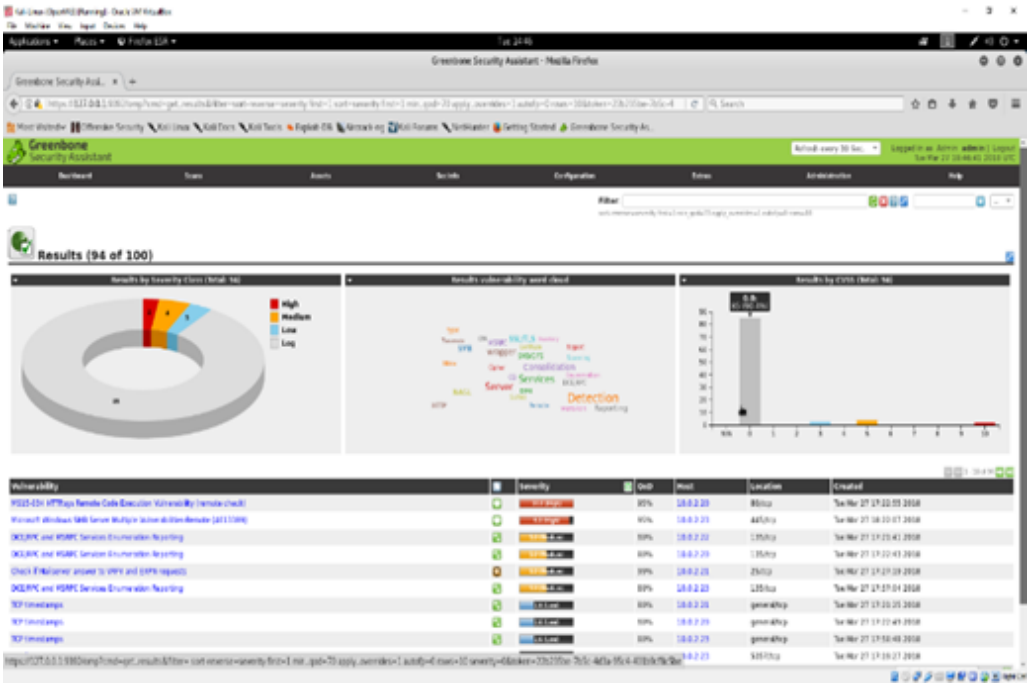


Fig 32. OpenVAS vulnerabilties results scanning the network

The first one was MS15-034, the same one found by Nessus on the Windows 2012 server and was also given a score of 10 and was presented in a very similar way

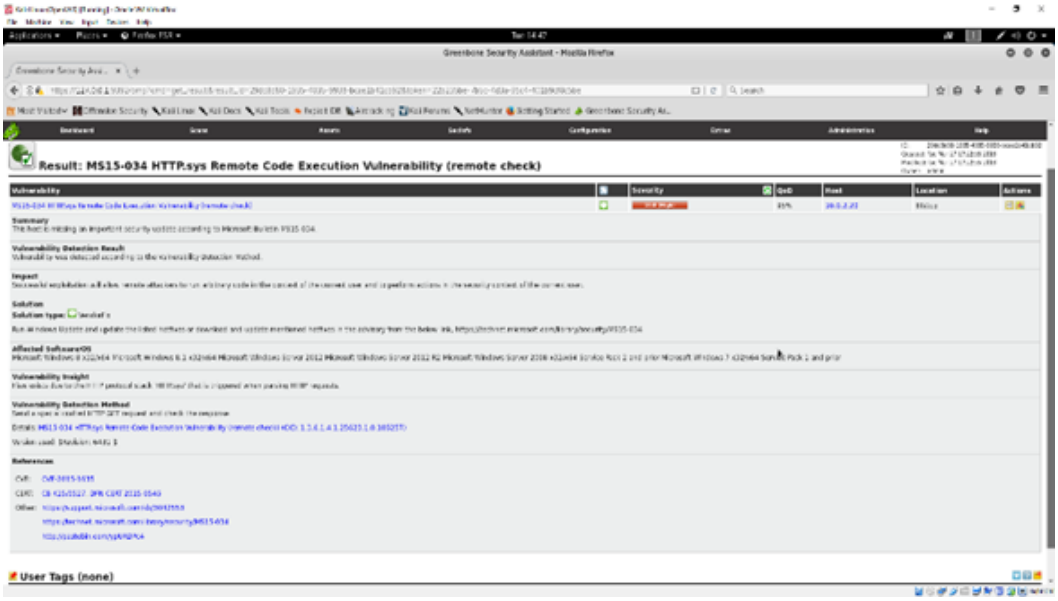


Fig 33. OpenVAS vulnerabilties results scanning the network analysis

On the windows 7 machine the same MS17-010 vulnerability was found and was given a 9.3 score (instead of 10 that Nessus gave it). In comparison to Nessus OpenVAS missed the MS11-030 critical vulnerability that was given a base score of 10 by Nessus as well as the warnings about SSL certificates on the Ubuntu server. The OpenVAS reports could be extracted as XML, HTML and PDF

3.3 Scanning the VM network with NSE

The NSE default scan can be initiated through Zenmap (the GUI for Nmap) by adding the " –nse default,safe" argument and a range of hosts to scan can be specified like the other scanners, in this case 10.0.2.20-23.

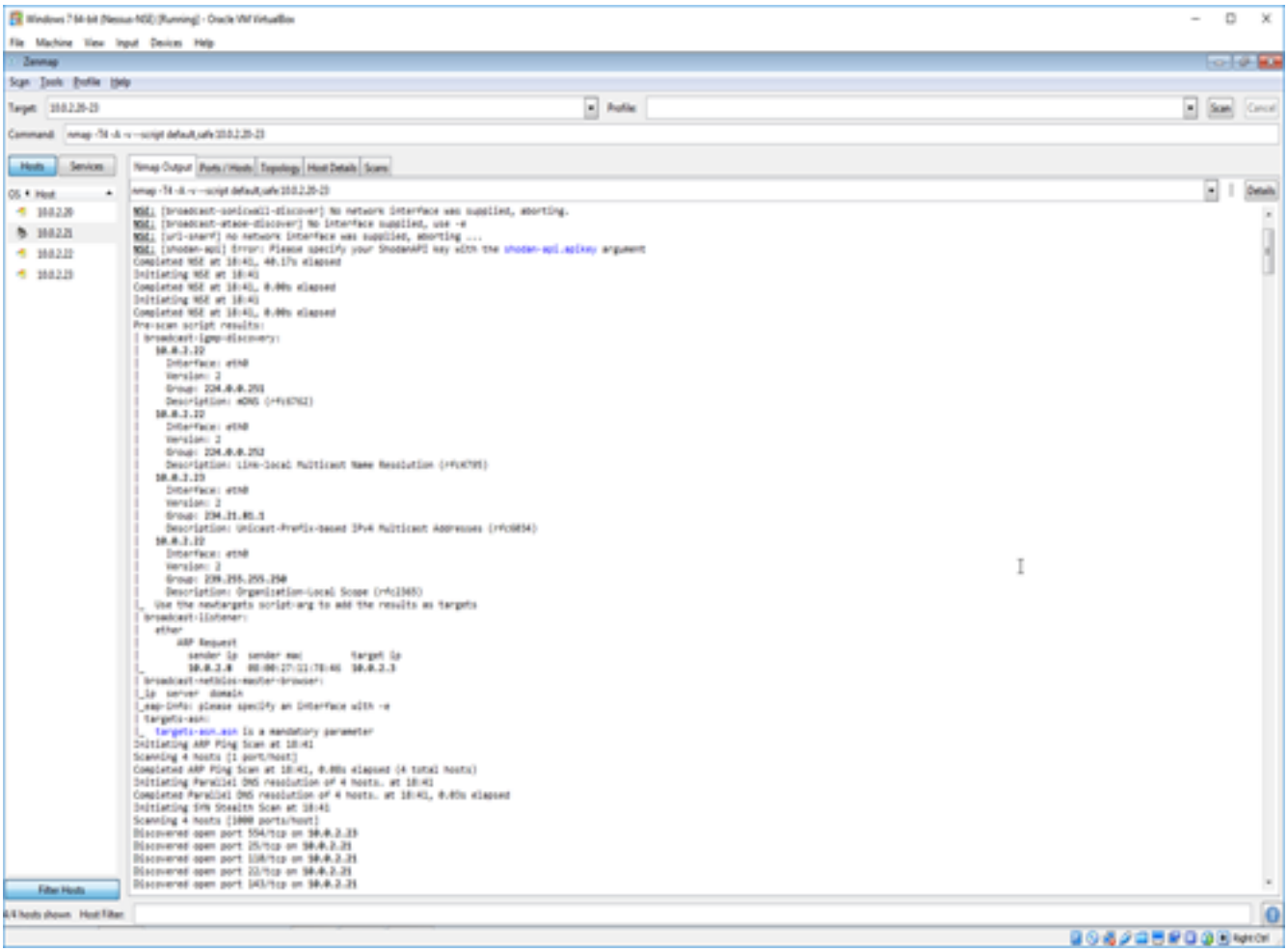


Fig 34. NSE vulnerabilties results scanning the experimental VM network

The scan on Zenmap completed in a bit less than 10 minutes, this was less than half the time it took Nessus.

```
Read data files from: C:\Program Files (x86)\Nmap

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 4 IP addresses (4 hosts up) scanned in 584.41 seconds

           Raw packets sent: 7171 (323.686KB) | Rcvd: 1151 (48.914KB)
```

While the report is less colorful than the other scanners through the UI the information about the network/hosts and ports/services is very clearly presented.

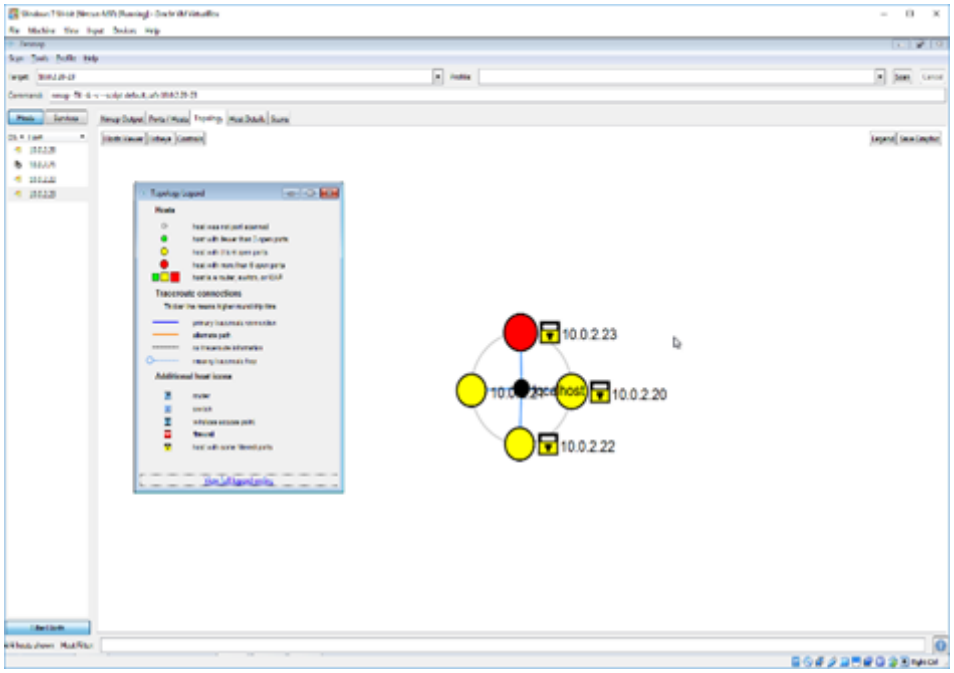


Fig 35. overview of the topology of the network

This is an overview of the topology of the Network
This is information about the host showing it's IP and OS

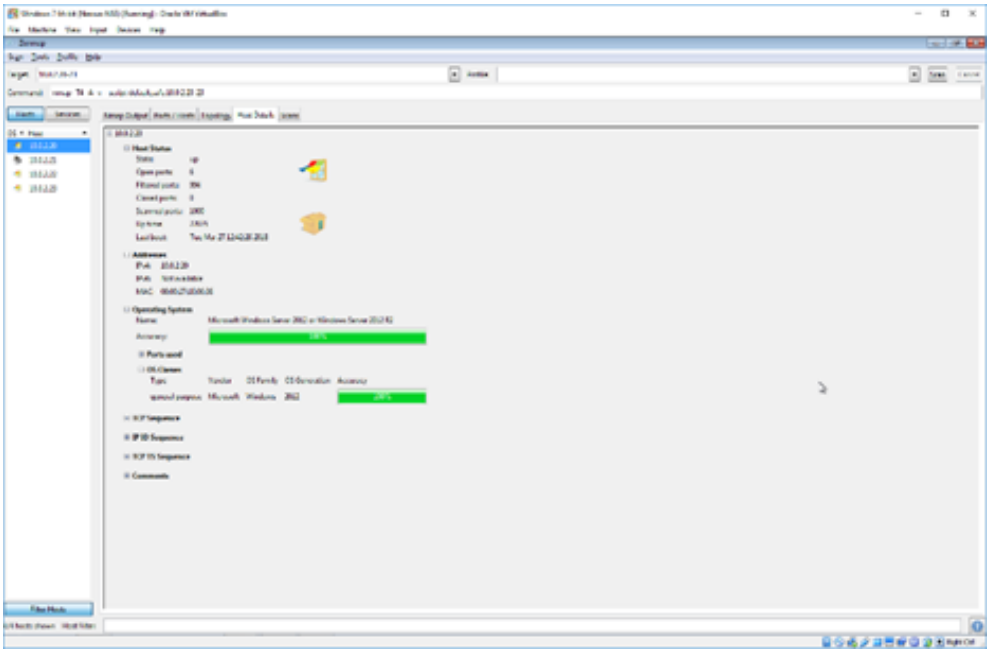


Fig 36. IP and OS configurations

As expected with Nmap's port scanning capabilities the information about open ports is very clearly presented.

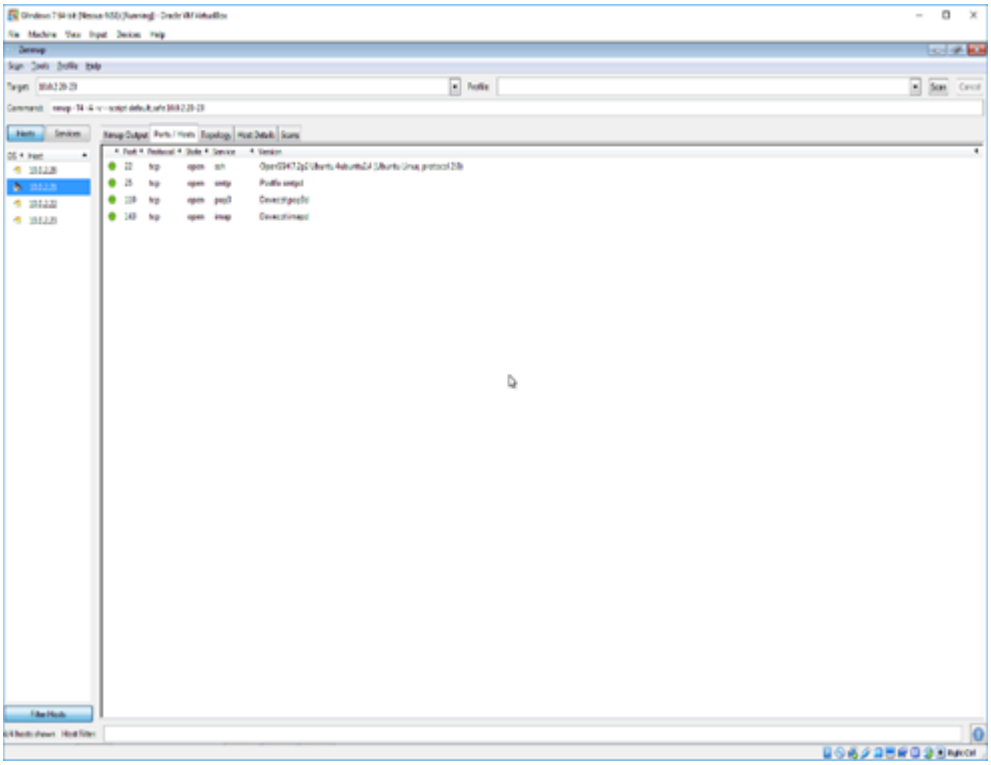


Fig 37. Nmap's port scanning capabilities report

As far as vulnerabilities go NSE is slightly less descriptive than the other scanners as it doesn't give a score, instead it gives a description and a CVE link (that contains the score).

The MS15-034 vulnerability was also found on the Windows 2012 server, a Risk factor was not given though even though the other scanners described it as critical.

```
| http-vuln-cve2015-1635:
|   VULNERABLE:
|   Remote Code Execution in HTTP.sys (MS15-034)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2015-1635
|       A remote code execution vulnerability exists in the HTTP protocol stack (HTTP.sys) that is
|       caused when HTTP.sys improperly parses specially crafted HTTP requests. An attacker who
|       successfully exploited this vulnerability could execute arbitrary code in the context of the System account.
|
|     Disclosure date: 2015-04-14
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1635
|_      https://technet.microsoft.com/en-us/library/security/ms15-034.aspx
```

Fig 38. NSE vulnerabilties results scanning the experimental VM network analysis

The MS17-010 vulnerability was found and was given a HIGH risk factor (the other two scanners also found the vulnerability and described it as CRITICAL with a 10 and 9.3 score.

Fig 39. NSE vulnerabilties results scanning the experimental VM network analysis (cont.)

It also found the SSL vulnerability on the Ubuntu server that OpenVAS didn't report on.

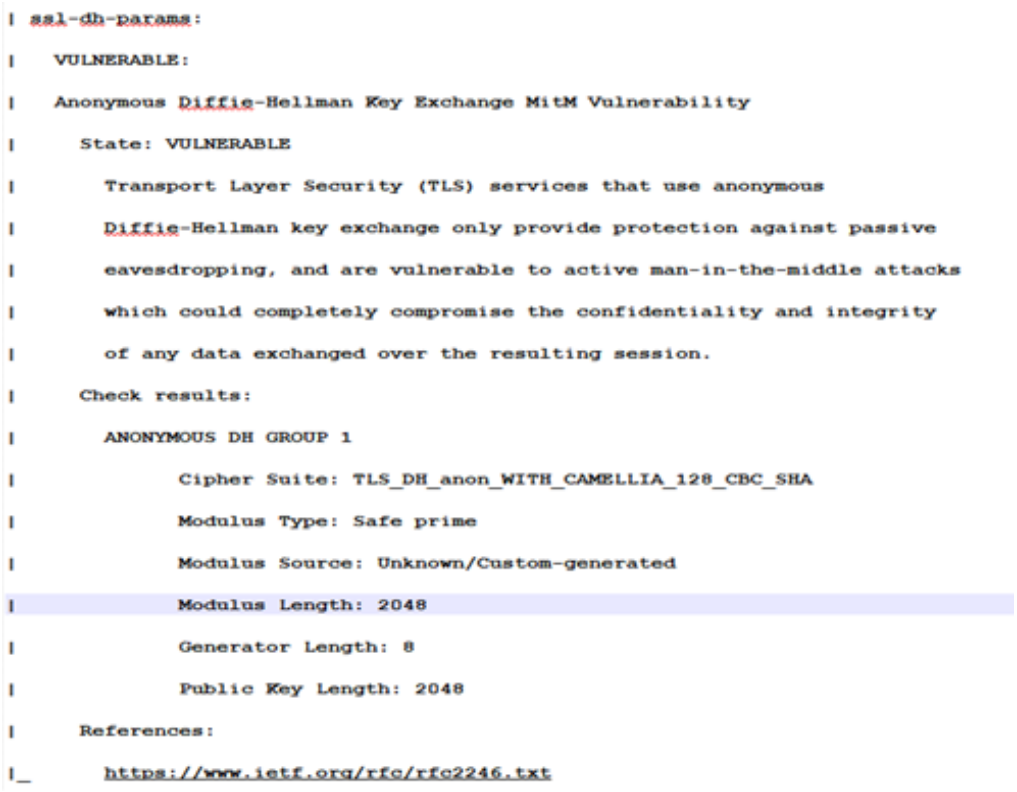


Fig 40. Nmap vulnerability report

The Nmap report can only be exported in XML format.

4 Knowledge Base Comparison

While presentation and the provided features are very important for any kind of software the quality of the Knowledge Base is arguably the most important aspect of any Vulnerability scanner. And while speed, reliability and the quality of the plugins is important the number of vulnerabilities covered is what provides the most coverage against attacks. The vendors usually

mention the number of plugins or checks when they are talking about their knowledge base but since a single plugin can refer to a number of vulnerabilities a more realistic way to figure out how many different vulnerabilities are covered is to check the CVE links included in their Knowledge base.

To perform this we used the plugin databases for Nessus, OpenVAS and Nmap that can be freely downloaded from Vulners.com. In order to process the vulnerability Knowledge Base for each scanner we used the Python scripting language to extract the unique CVEs that each database covers and placed them in sets (which are unordered lists in Python that contain unique objects). From these sets we extracted the following information about how many unique Vulnerabilities each database covers.

The downloaded Knowledge Base is included in a Zip file as a JSON file. A JSON file (JavaScript Object Notation) is a way to store information in a human readable collection of organized data that can be accessed in a logical manner.

First, to import the data for the Nessus zip archive we can use the following commands:

```
Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from zipfile import ZipFile
>>> archive = ZipFile("nessus.json.zip")
>>> archived_file = archive.open(archive.namelist()[0])
>>> archive_content = archived_file.read()
>>> archived_file.close()
>>>
```

This loads the JSON objects included in the downloaded archive into the object vulners_objects and creates the nessus_cves set that contains a list of dictionaries for each object (ID, title and CVE list). This set (which is an unordered list in Python that contains unique objects) contains the CVE references contained in the Knowledge base. It is important that those are unique to avoid duplicate CVE references that would inflate the number of unique vulnerabilities covered by each Knowledge base.

```
>>> import json
>>> vulners_objects = json.loads(archive_content)
>>> nessus_cves = set()
>>> nessus_objects = dict()
>>> for obj in vulners_objects:
...     nessus_objects[obj["_id"]] = obj
...     for cve in set(obj["_source"]["cvelist"]):
...         nessus_cves.add(cve)
```

We can see this information contained by asking to print the ID, title or cvelist of a random plugin.

```
>>> print(vulners_objects[39000]["_id"])
FEDORA_2015-9601.NASL
>>> print(vulners_objects[39000]["_source"]["title"])
Fedora 22 : qemu-2.3.0-5.fc22 (2015-9601)
>>> print(vulners_objects[39000]["_source"]["cvelist"])
['CVE-2015-4037']
```

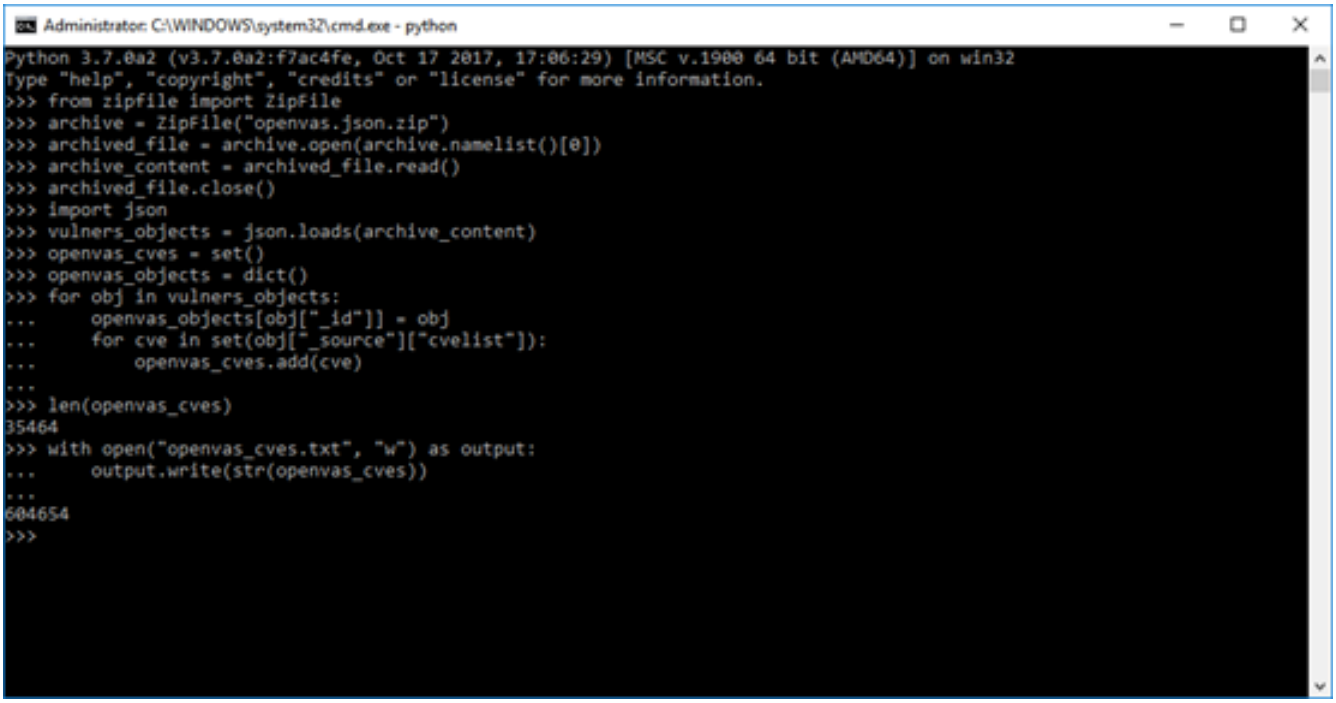
This information can be exported in a text file in a format that can be easily imported to a software like excel or a database like access containing just the list of available CVEs for the downloaded Nessus Knowledge Base.

```
>>> with open("nessus_cves.txt", "w") as output:
...     output.write(str(nessus_cves))
...
699301
```

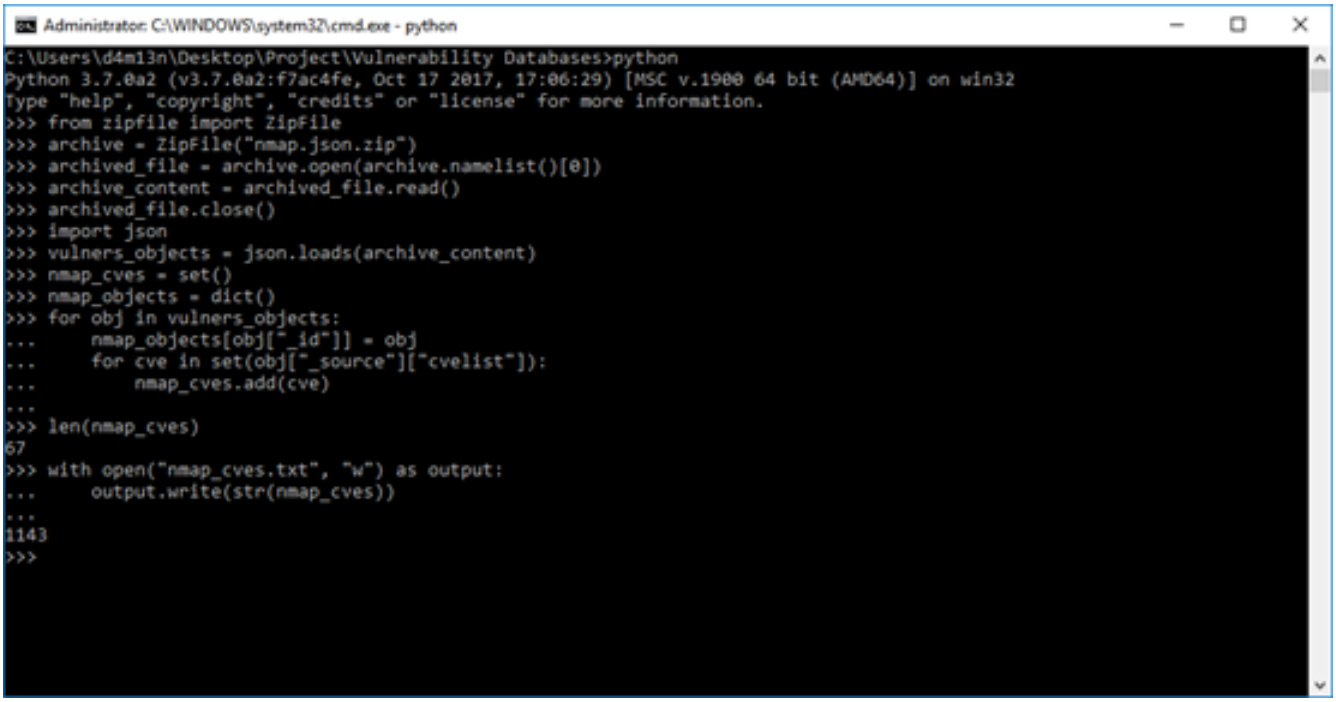
In order now to retrieve the number of unique CVEs we can use another Python command "len" which returns the length of the object that as we said contains only unique CVEs.

```
>>> len(nessus_cves)
41030
```

From this we can extract that the Nessus Vulnerability Knowledge Base downloaded from Vulners.com detects 41.030 different vulnerabilities. The same process can be repeated for OpenVAS



The OpenVAS Vulnerability Knowledge Base downloaded from Vulners.com detects 35.464 different vulnerabilities.
As well as Nmap:



The Nmap Vulnerability Knowledge Base downloaded from Vulners.com detects 35.464 different vulnerabilities.

Comparing the Knowledge bases we can see that Nessus covers more than 5000 different vulnerabilities than OpenVAS while the Nmap scripting engine only covers a total of 67 different Vulnerabilities which is to be expected from the much lower number of scripts it has available for vulnerability scanning.

Of course this is not an error free comparison because of the possibility of missing CVE links or not inclusion of old vulnerabilities or vulnerabilities for specific software.

Even though one database contains more CVEs than the other it doesn't mean it includes all the CVEs that the smaller database contains. Using the created sets of unique CVEs we can use another operation of Python sets ".difference" which compares two sets and returns a new set with elements that exist in the first but not the second.

```
>>> NessusMinusOpenVAS=set(nessus_cves).difference(openvas_cves)
>>> len(NessusMinusOpenVAS)
10760
```

Even though Nessus covers 5,566 more unique vulnerabilities than OpenVAS while comparing the two the Nessus database contains 10,760 CVEs that the one of OpenVAS doesn't.

```
>>> OpenVASminusNessus=set(openvas_cves).difference(nessus_cves)
>>> len(OpenVASminusNessus)
5194
```

So with this in mind it is expected that OpenVAS covers 5,194 vulnerabilities that Nessus doesn't. This means that even though OpenVAS covers less CVEs there might be some vulnerabilities that will be detected by OpenVAS but not Nessus.

```
>>> NessusMinusNmap=set(nessus_cves).difference(nmap_cves)
>>> len(NessusMinusNmap)
40967
>>> OpenVASMinusNmap=set(openvas_cves).difference(nmap_cves)
>>> len(OpenVASMinusNmap)
35405
>>> NmapMinusNessus=set(nmap_cves).difference(nessus_cves)
>>> len(NmapMinusNessus)
4
>>> NmapMinusOpenVAS=set(nmap_cves).difference(openvas_cves)
>>> len(NmapMinusOpenVAS)
8
>>>
```

Table 1. Unique vulnerabilities covered

Vulnerability scanner	Unique CVEs
Nessus	41030
OpenVAS	35464
NSE	67

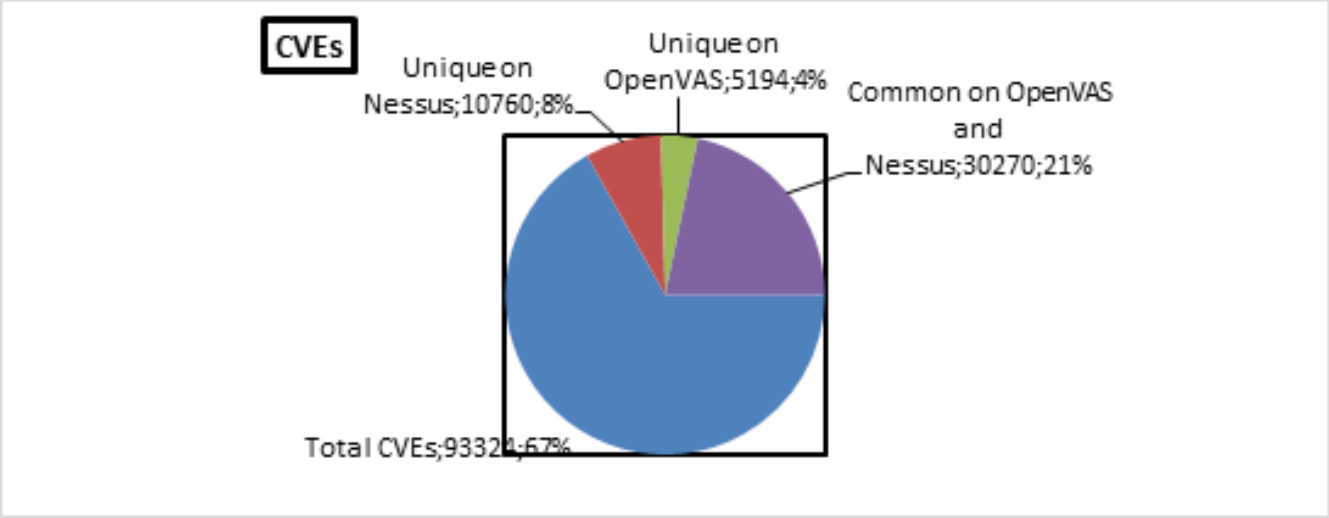


Fig 41. Unique CVEs on each database

Even though Nessus covers 5,566 more unique vulnerabilities than OpenVAS while comparing the two the Nessus database contains 10,760 CVEs that the one of OpenVAS doesn't. As a result OpenVAS covers 5,194 vulnerabilities that Nessus doesn't. This means that even though OpenVAS covers less CVEs there might be some vulnerabilities that will be detected by OpenVAS but not Nessus. Even Nmap that covers a much smaller number of vulnerabilities than both of the other tools could potentially detect 4 and 8 vulnerabilities that Nessus or OpenVAS respectively would not.

Looking at the overall number of existing CVE IDs (93324) even the numbers of Nessus and OpenVAS seem small but in reality the CVEs that they are not detected at this point by a scanner could just be that they are old or very specific to certain software or devices and due to these reasons the differences on the Nessus and OpenVAS databases can be much smaller that it seems.

5 Performing A Risk Assessment

In order to assess the usefulness of each scanner in evaluating the overall relative security of a network it is important to define what the requirements for such a risk assessment are. Then we need to investigate what data we can get from these vulnerability scanners and how we can use it to perform a Risk Assessment of the system. According to the NIST (National Institute of Standards and Technology) Risk Management Guide for Information Technology Systems "Risk is a function of the likelihood of a given threat-source's exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organization".[(2)]

The Risk Assessment Methodology can be described in 9 steps and in the following section we'll be examining how the three vulnerability scanners we are comparing (Nessus, OpenVAS and NSE) can be used to gather and present information in regards to each of these steps.

Step 1. System Characterization

The first step before starting to assess risk of a system is to collect system related information like Hardware, Software, Interfaces, Topology, Data and Information, and the People that use and support the system as well as the System mission (the purpose of the system and the processes that are performed by it).

There is also information used to determine the actual risk value in relation to the data like System and data criticality (the value of the data) as well as the System and data sensitivity (the level of protection required to maintain the system and data integrity, confidentiality and availability). Some of this information cannot be gathered by an automated computer software on an unknown system like the information on the people that use the system, its purpose and the criticality and sensitivity of data. The recommended techniques to gather this information are through human interaction (Questionnaires and Interviews) as well as Document Reviews and could vary on different systems and organizations but still in a proposed software solution that uses online questionnaires or other data gathering techniques this information could be parsed directly to the software through a standardized interface.

What can be performed by an automated scanning tool and is in fact mostly performed by using one of those (network mappers) is identifying the hosts and the services that they are running on the system. All three of the tools can be used for Host and Service Discovery and this information is presented on their final reports.

Step 2. Threat Identification

Threat is described as the potential for a particular threat-source to successfully exploit a vulnerability in a way that can harm the system.[(2)]

Even though sources and motivation for their threat actions cannot be identified by a vulnerability scanner and should be investigated specifically for each organization and its data by the management and the IT professionals, knowing what vulnerabilities exist and where in the system can provide with an indication of who could potentially seek to exploit them and what would be their motivation.

Step 3. Vulnerability identification

Vulnerability is described as a flaw or weakness in system security procedures that could be exercised and result in a security breach or violation of the system's security policy.[(2)]

Again in the larger scope of an organization this could include the physical and environmental security of the IT system as well as the personnel that are out of the scope of vulnerability scanning software but what they can definitely be used for (and is as expected their main use in IT security), is to identify software vulnerabilities in the system like known vulnerable services, missing patches and dangerous configurations that could be exercised by the potential threat-sources.

Step 4. Control Analysis

This step is about analyzing the controls that exist or are planned to minimize the probability of a threat-source exercising a system vulnerability. These controls can be Preventive that prevent attempts to violate security (firewalls, encryption, and authentication) or Detective that warn of violations or attempted violations of the security measures.

The vulnerability scanners can be can be used to notify on the absence of these types of technical controls.

Step 5. Likelihood Determination

A likelihood rating indicates the probability that a vulnerability may be exercised by a threat-source and according to NIST it can be categorized into three likelihood levels.[(2)]

A similar way or rating vulnerabilities is used in Nessus, OpenVAS and NSE, the Common Vulnerability Scoring System that assigns a CVSS score on each vulnerability that the scanner uses as a method to warn the user about the vulnerabilities that present the bigger risk and should be attended to first.

Nessus sorts the vulnerabilities in the order of Critical, High, Medium and Low. Each vulnerability is accompanied with Risk information that apart from the mentioned Risk Factor has a CVSS Base Score and CVSS temporal score. The difference with the temporal score is that it can change overtime based on factors like exploitability and remediation level.

Table 2. Likelihood determination

Likelihood level	Likelihood definition
High	The threat-source is highly motivated and sufficiently capable, and controls to prevent the vulnerability from being exercised are ineffective
Medium	The threat-source is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability
Low	The threat-source lacks motivation or capability, or controls are in place to prevent, or at least significantly impede, the vulnerability from being exercised.

NIST likelihood determination categorization

OpenVAS also uses the CVSS scoring system in a similar way to present and sort the vulnerability exploitability risk information. One difference with Nessus is the Risk Factors are High, Medium and Low.

NSE provides with the same Risk Factor characterization (High, Medium and Low) in its reports and while it doesn't provide with a CVSS score directly this information can be retrieved from the CVE link it provides.

It should be noted that while these scores are indicative of how urgently a vulnerability should be looked at they don't tell the whole story on the motivation of a threat-source to exploit this vulnerability but it makes sense that from the viewpoint of a threat-source highly exploitable vulnerabilities would be a priority when they are looking to harm the system.

Step 6. Impact Analysis

Determining the impact resulting from a successful threat exercise of a vulnerability is a major factor in measuring the level or risk.

While the CVSS score can indicate the severity of a vulnerability being exploited in the system it doesn't account for the importance and the sensitivity of the host or the data that could be affected and how this could impact the system's mission and the organization overall.

Impact can be assigned by the owners/managers based on three security goals: Loss of Integrity (information must be protected from improper modification), Loss of Availability (a system or information being unavailable to users could affect the operations of the organization), Loss of Confidentiality (protection from unauthorized disclosure of information). While sometimes impact can be directly translated to loss in revenue or cost of repairs other impacts can be easier qualified in terms of high, medium and low impacts (similarly to the Risk Factors) according to this table recommended by NIST.[(2)]

Table 3. Impact analysis

Magnitude of Impact	Impact Definition
High	Exercise of the vulnerability (1) may result in the highly costly loss of major tangible assets or resources (2) may significantly violate, harm or impede an organization's mission, reputation or interest; or (3) may result in human death or serious injury
Medium	Exercise of the vulnerability (1) may result in the costly loss of tangible assets or resources (2) may significantly violate, harm, or impede an organization's mission, reputation, or interest; or (3) may result in human injury
Low	Exercise of the vulnerability (1) may result in loss of some tangible assets or resources or (2) may noticeably affect an organization's mission, reputation, or interest.

NIST impact analysis recommendation

Step 7. Risk determination

Table 4. Risk determination

Threat Likelihood	Impact		
	Low (10)	Medium(50)	High(100)
High (1.0)	10x1.0=10	50x10.0=50	100x1.0=100
Medium (0.5)	10x0.5=5	50x0.5=25	100x0.5=50
Low (0.1)	10x0.1=1	50x0.1=5	100x0.1=10

a. Risk Scale : High (>50 to 100) Medium (>10 to 50) Low (1 to 10)

In order to assess the level of risk for a particular vulnerability a risk-level matrix can be developed by multiplying the ratings assigned for likelihood and impact of a threat that where determined in steps 5 and 6. Assigning this ratings can be subjective

but an in Table 4 we can see an example of a 3x3 probability matrix provided by NIST.[2]

Using this matrix each vulnerability can be assigned a Risk Level depending on the probability that it will be exploited and the negative impact that it would have on the system. These risk levels can be used to aid the decisions of managers in regards to how to act about each vulnerability like the example NIST recommendations shown in Table 5 .

Table 5. Risk description and necessary actions

Risk Level	Risk description and necessary actions
High	If an observation or finding is evaluated as high risk, there is a strong need for corrective measures. An existing system may continue to operate, but a corrective action plan must be put in place as soon as possible
Medium	If an observations is rated as medium risk, corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time.
Low	If an observation is described as low risk, the system's DAA must determine whether corrective actions are still required or decide to accept the risk

NIST recommendations for Risk descriptions andnecessary actions

Step 8. Control Recommendations

The decision to minimize or eliminate risks is a management decision too. Because this process is usually associated with a cost whether it is financial, effort or operational impact the Risk Level is an important factor to decided which Risks should be eliminated and at which order.

While the vulnerability scanners can't help with the decision in regards to the cost the Risk Factor/ CVSS score does help directly to determine the overall vulnerability risk and most of the times they offer technical recommendations on how to eliminate this risk.

Step 9. Results Determination

The final step of Risk assessment is collecting all the information gathered by the previous steps and presenting them in a report in order to help the managers/owners make decisions.

The basic structure of this report is not very different than the ones vulnerability scanners produce. There is still information about the system (hosts, services, interfaces and the addition of users and data), the vulnerabilities found are sorted by a risk factor (with the addition of impact in its calculation in a risk assessment) and a solution is proposed for each vulnerability in order to control it (in a risk assessment the solution could be do nothing if it is not cost effective). Where a risk assessment report differs is that instead of being aimed towards IT professionals it is presented in a way that managers can understand them and make decisions without being technically proficient themselves.

6 Scripting Capabilities

Since all three vulnerability scanners we are examining offer the ability to create custom tests and reports using powerful scripting languages (NASL for Nessus and OpenVAS and LUA for NSE) it is worth examining the capabilities of these scripting languages to create custom scripts that receive external input, process it along with existing vulnerability scanning tests and produce a custom report that is understandable from non-technical managers following the steps of risk assessment described earlier.

Nessus Attack Scripting Language (Nessus and OpenVAS)

Nessus Attack Scripting Language (NASL) was developed for the Nessus security scanner and since OpenVAS branched out of the open source version of Nessus its scripts are built on NASL too.

Each host is associated to an internal knowledge base, which contains all the information gathered by the tests during the scan. This enables the tests to exchange info and save time and resources. The fact that we can store our own values in the KB means that values like impact could be manually input, stored and shared between scripts as well. These values could be combined with internal variables on the custom scripts. The processing of results as well as the final presentation of the report can be customized in Nessus and OpenVAS using NASL scripts. If a cumulative report that combined all the tests reports was required (for example for our purposes to calculate a risk factor by multiplying probability with impact), the Knowledge Base could be used by a NASL script that retrieves information and combines them conditionally. What makes the report string convenient to be used is the fact that it doesn't have to be created all at once. Strings can be manipulated like in C for example where strings can be added in different order. So different functions of the script can create their own strings based on their own conditionals and results and those can be summed up in the end to form the report string in the desired order.

A way to make feedback more specialized for each organization and useful to any manager could be achieved through Nessus or OpenVAS is by creating a superscript (a combination of NASL scripts) that combines the important/relevant tests for a system

with info specific to the system or organization by the user to provide cumulative feedback about the security of the system in the form of a Risk Assessment.

NASL allows developers to easily forge IP packets or send regular packets while providing functions to make the tests of web and ftp servers easier to write. While doing that it guarantees security in the sense that it will not send any packet to the host other than the target host and it will not execute any commands on the local system.

While it is not as fast or powerful as other scripting languages like Python or LUA for more complicated scripts it has the great benefit of being secure for its purpose (security tests) while these other languages aren't. You can't write Trojans with NASL that open connections to third parties and authenticate as Nessus users and send important files (like passwd) for example.

NASL is also less memory hungry than these other languages and self-sufficient in the sense that you don't need to install any other packages to write a security test.

There is the fact of course that it is a specialized language that has to be learned in order to write security tests for Nessus or OpenVAS but users that want to develop their own security tests must keep in mind that :

- NASL is optimized for Nessus. Writing a Nessus test in this language is fast
- NASL has a lot of things in common with C, so it's easy to learn
- NASL produces secure and easily sharable security tests
- NASL produces portable and easily modified security tests. The same functions are used to do the same things through different OS.

The biggest limitations of NASL at this point are:

- Structures. They are not supported'
- correct debugger. However there is a stand-alone interpreter 'nasl'

Being very similar to C in structure NASL allows for the usual things like variables, numbers and strings, for and while loops, user defined functions, standard arithmetic and binary operators.

It also comes with functions to manipulate sockets, packets and other handy utility functions that make network operations easier like getting the IP and name of a host or checking if a port is open.

Each host is associated to an internal knowledge base, which contains all the information gathered by the tests during the scan. The security tests are encouraged to read it and to contribute to it. The status of the ports, for instance, is in fact written somewhere in the knowledge base. This enables the tests to exchange info and save time and resources.

Basically, there are two functions regarding the knowledge base. The get_kb_item(<name>) function will return the value of the knowledge base item <name>. This function is anonymous. The function set_kb_item(name:<name>, value:<value>) will mark the new item <name> of value <value> in the knowledge base.

Using the various functions of the knowledge base the user should make sure that the script is as lazy as possible - that is, it must not do something that another script has already done.

NASL is a rather straight forward scripting language and similar to C in many aspects while at the same time it is secure and specialized to perform network functions with the purpose to test a system's security against specific attacks or vulnerabilities.

There are thousands of NASL scripts available (for both Nessus and OpenVAS) in the NASL distribution for the most common systems and vulnerabilities but it is also easy enough to develop custom tests for those that don't exist (like custom company software).

In order to present the results to the user through Nessus NASL scripts use functions like security_warning(), while each script can use the Knowledge Base to store information retrieved from the tests it performed. So this means the tests, the processing of the results and the final presentation of the results can be customized in Nessus and OpenVAS using NASL scripts. NASL scripts can provide feedback in the form of plain text and hypertext to the user and that's what Nessus and OpenVAS both use.

While the description string is static the rest of the report doesn't have to be. While forming the strings variables can be used (including Knowledge Base items) and also the content of the Strings can be decided through conditionals (for example if this variable is true use this string else use another one).

What makes the report string convenient to be used is the fact that it doesn't have to be created all at once. Strings can be manipulated like in C for example where strings can be added in different order. So different functions of the script can create their own strings based on their own conditionals and results and those can be summed up in the end to form the report string in the desired order.

For example a script can create different strings like string1,string2,string3 and then create the final report by combining them : "report = string1 + string2 + string 3". The same way those substrings can be updated "string1 = string1 + "\n This is an addition to String1".

The report string of the script is ultimately the end result of the script as far as the user can see presenting the results in a text form with possible hyperlinks.

The UI of both Nessus and OpenVAS is simple and user friendly, an important enhancement to its function as a security tool would be making the feedback more specialised for each organization and readable by the manager, in effect more useful as a metric for the total security of the system.

Lua scripting language (NSE)

NSE scripts are written in the embedded Lua programming language.

Lua is described by its creators as a powerful, efficient, lightweight and embeddable scripting language that supports procedural programming, object - oriented programming, functional programming, data-driven programming and data description.

Some of the main advantages of Lua are:

- Lua is a proven, robust language used in many industrial applications and games
- Lua is fast, in fact several benchmarks show it as the fastest scripting language
- Lua is portable, it build out-of-the-box in all platforms that have a standard C compiler
- Lua is powerful, yet simple for example even if it's not a pure object-oriented language it provides meta-mechanisms for implementing classes and inheritance
- Lua is small, source code and documentation is 1.1M uncompressed
- Lua is free, it is also open-source and can be used for any purpose including commercial purposes at no cost

Lua supports data structures such as Boolean values, numbers and strings as well as data structures such as arrays, sets, lists and records using its single native data structure, the table, which is similar to an associative array in PHP.

Its syntax is simple, someone that knows C, Python or PHP can easily learn to use Lua. It supports functions, four types of loops, tables as data structures, metatables that contain data that acts as source code or metadata and even though it doesn't have a built-in concept of classes it can achieve object oriented programming using first class functions (passing functions as arguments, returning them as values for other functions and assigning them to variables or storing them into data structures) and storing them into tables. As it is intended to be embedded into other application it provides a C API for this purpose.

NSE being a part of Nmap adds a versatile and powerful library collection that allows easy interaction with most major network services and protocols and can use Nmap's powerful and fast host and port scanning engine.

The NSE "vulns" library that was developed in order to standardize vulnerability presentation and easy vulnerability management can be very useful in order to present a vulnerability report in a standardized way. Using the 'make_output" function of this library a report is automatically generated where we can give the vulnerability a title and CVE ID.

Using the vulns library we can aggregate and report results across multiple hosts. This is done by saving vulnerabilities in nmap's registry which is persistent across multiple host scans and later format this combined output as a postrule script which can then be further formatted.

Where NSE could potentially edge out Nessus or OpenVAS in regards as being used as part of a Risk Assessment tool is the embeddability of Lua in other applications if it is decided that using a different software as a User Interface for manager input (like risk impact) or to process and present the results is a better solution with NSE being just a scanner in the back end.

7 Using a Vulnerability Scanner for Risk Assessment

We identified in the previous sections that performing a Risk Assessment involves more data than what a vulnerability scanner usually can retrieve. This is in all cases data that cannot be retrieved automatically as it is more specific to each organization and is usually acquired by questioning the managers.

As we can see in a simplified solution two additional functions to the scanner's normal operations need to be performed (the calculation of risk and the compilation of the custom final report) as well as one important extra input to for the risk determination calculations which is the magnitude of impact associated with each host.

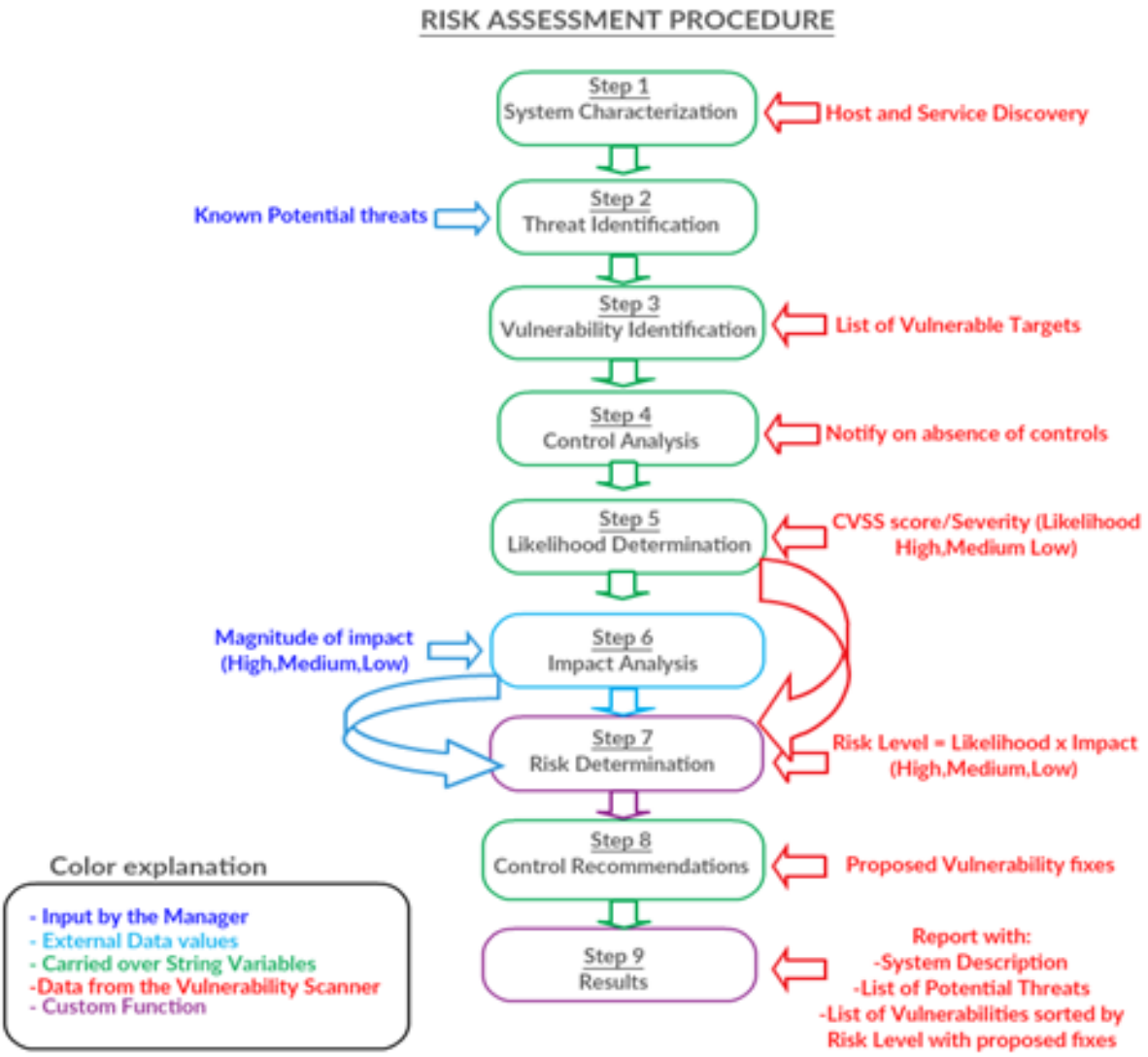


Fig 42. Risk assessment with a vulnerability scanner

8 Conclusion

The main conclusion that can be drawn from this comparison is that all three of these vulnerability scanners might be suitable candidates for the purpose of creating a Risk Assessment tool.

Had it remained free and open source Nessus would be the overall best choice due to its market dominance, available documentation and vulnerability database. OpenVAS can do most of what Nessus can and the real difference in vulnerability detection is not as big as the plugin numbers suggest as we examined earlier. NSE is very interesting due to the other features that come along with Nmap and its ability to penetration testing as well as LUA being an overall more powerful scripting language as well as better documented than NASL. The limited vulnerability database makes it a poor choice for a tool that works on already existing scripts but it could be viable as a more personalized tool for a specific network. It is also definitely worth it to do further research on how a stand-alone risk assessment tool can work in collaboration with all these vulnerability scanners to collect data from their reports and even aggregate the results of different vulnerability scanners and other security tools to provide a more complete risk assessment report. Overall, this work extending the preliminary results of[1], provides the complete setup and analysis of the proposed virtual machine based framework towards a comprehensive risk assessment tool for small to medium

size enterprises.

References

1) Chalvatzis I, Karras DA, Papademetriou RC. Evaluation of security vulnerability scanners for small and medium enterprises business networks resilience towards risk assessment. In: and others, editor. International Conference on Artificial Intelligence and Computer Applications. IEEE. 2019;p. 52–58.
2) Manson S, Anderson D. Cybersecurity for Protection and Control Systems: An Overview of Proven Design Solutions. Institute of Electrical and Electronics Engineers (IEEE). 2019. doi:10.1109/mias.2018.2875175.
3) Humayed A, Lin J, Li F, Luo B. Cyber-Physical Systems Security—A Survey. *IEEE Internet of Things Journal*. 2017;4(6):1802–1831. doi:10.1109/jiot.2017.2703172.
4) Furnell SM, Clarke N, Werlinger R, Muldner K, Hawkey K, Beznosov K. Preparation, detection, and analysis: the diagnostic work of IT security incident response. *Information Management & Computer Security*. 2010.
5) Holm H, Sommestad T, Almroth J, Persson M. A quantitative evaluation of vulnerability scanning. . *Information Management & Computer Security*. 2011. doi:10.1108/09685221111173058.
6) Holm H. Performance of automated network vulnerability scanning at remediating security issues. . *Computers & Security*. 2012;31:164–175.
7) Badawy MA, El-Fishawy N, Elshakankiry O. Vulnerability scanners capabilities for detecting windows missed patches: Comparative study. In: International Conference on Security of Information and Communication Networks. Berlin, Heidelberg. Springer. 2013;p. 185–195.
8) Tripathy BK. Risk Assessment in IT Infrastructure. In: Ethics, Laws, and Policies for Privacy, Security, and Liability. 2020.