

Evaluation of Security Vulnerability Scanners for Small and Medium Enterprises Business Networks Resilience towards Risk Assessment

Ilias Chalvatzis

University of Portsmouth, UK, School of Engineering, Faculty of Technology, Anglesea Building, Anglesea Road, Portsmouth (UK) PO1 3DJ

Dimitrios A. Karras

Stereia Hellas Institute of Technology, Greece, Automation Dept., Psachna, Evoia, Hellas (Greece) P.C. 34400, dakarras@teiste.gr dimitrios.karras@gmail.com

Rallis C. Papademetriou

University of Portsmouth, UK, School of Engineering, Faculty of Technology, Anglesea Building, Anglesea Road, Portsmouth (UK) PO1 3DJ

Abstract—Risk Assessment has been identified 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 virtual machine based framework for testing the performance of vulnerability scanners applied to such enterprises, focused to small and medium size ones. Moreover, the purpose of this paper is to compare three of the most well-known free vulnerability scanning solutions (Nessus, OpenVAS, Nmap Scripting Engine) in regards to how they can be used to automate the process of Risk Assessment in an organization, based on the herein presented experimental evaluation framework involving virtual machine testing.

Keywords—Vulnerability Scanning, Risk Assessment, Nessus, OpenVAS, Nmap Scripting Engine

I. INTRODUCTION

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

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]

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.

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.

The aim of this report is to analyze the available vulnerability scanners and decide who would be the ideal as the base to create such a stochastic model and based on its features how this could potentially be achieved. Although, there are other attempts in the literature ([3], [4], [5]) to provide assessments of vulnerability scanners based on their features and characteristics, offering relevant comparison frameworks, the contribution of this paper lies in the provision of a suitable evaluation framework towards emulating business environment risk assessment. It is based on virtual machine testing and focuses on emulating the business environment of a small to medium size enterprise.

II. 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.

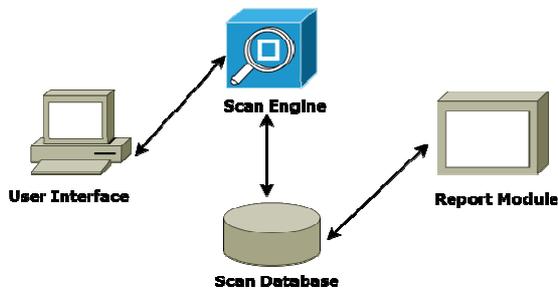


Fig. 1. Vulnerability scanner architecture

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.

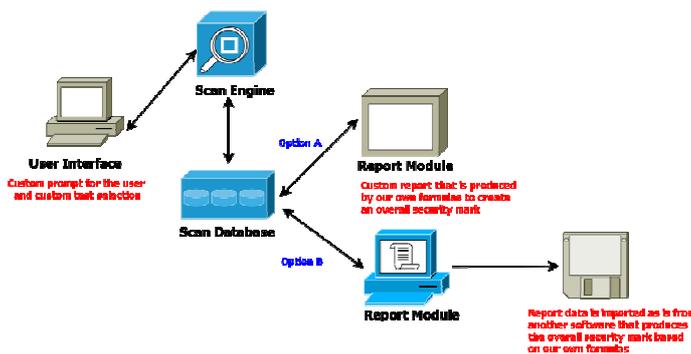


Fig. 2. Using a Vulnerability scanner to produce an overall security mark

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.

A. 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.

B. 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.

C. 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.

III. VIRTUAL MACHINE TESTING

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.

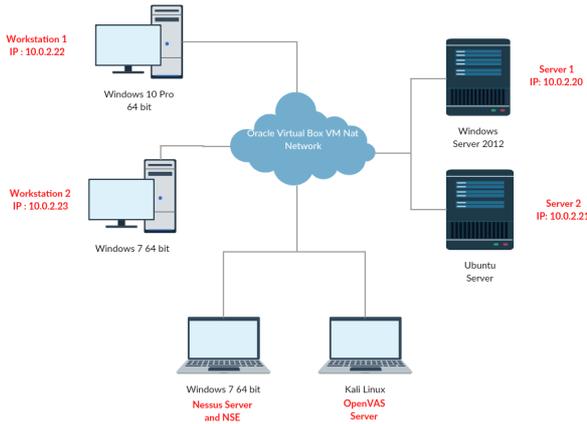
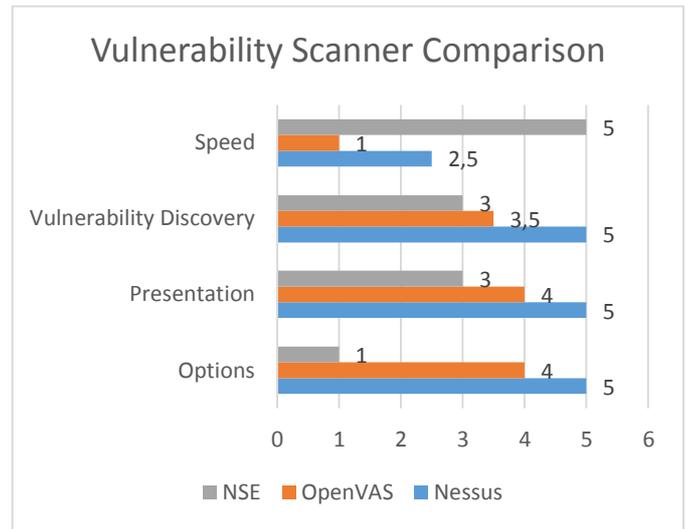


Fig. 3. Virtual Machines setup for testing the Vulnerability Scanners

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.



IV. 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.

TABLE I. UNIQUE VULNERABILITIES COVERED

Vulnerability Scanner	Unique CVEs
Nessus	41030
OpenVAS	35464
NSE	67

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.

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 used 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.

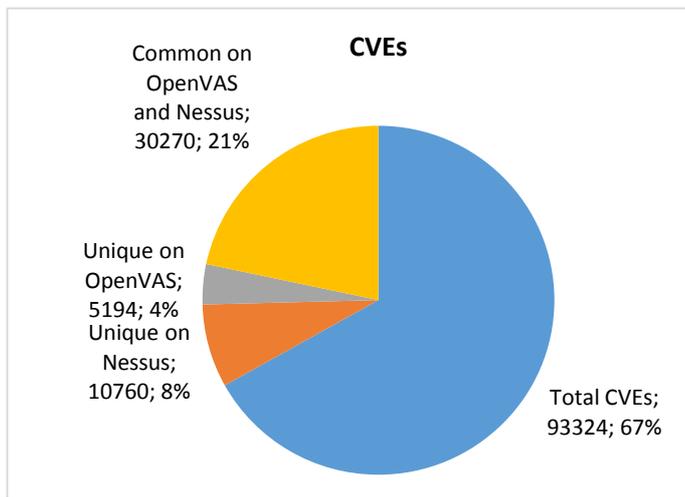


Fig. 4. Unique CVEs on each database

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.

V. 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 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]

TABLE II. 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.

^a. NIST likelihood determination categorization

A similar way of 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.

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 III. 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.

^b. NIST impact analysis recommendation

Step 7. Risk Determination

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 were determined in steps 5 and 6. Assigning this ratings can be subjective but in table IV we can see an example of a 3x3 probability matrix provided by NIST. [2]

TABLE IV. RISK DETERMINATION

Threat Likelihood	Impact		
	Low (10)	Medium(50)	High(100)
High (1.0)	10x1.0=10	50x1.0=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)

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 V.

TABLE V. 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

^b. NIST recommendations for Risk descriptions and necessary 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.

VI. 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.

A. 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.

B. Lua scripting language (NSE)

NSE scripts are written in the embedded Lua programming language. 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 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 but the lack of vulnerability checks at this point makes it a less appropriate solution for a universal tool.

VII. 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.

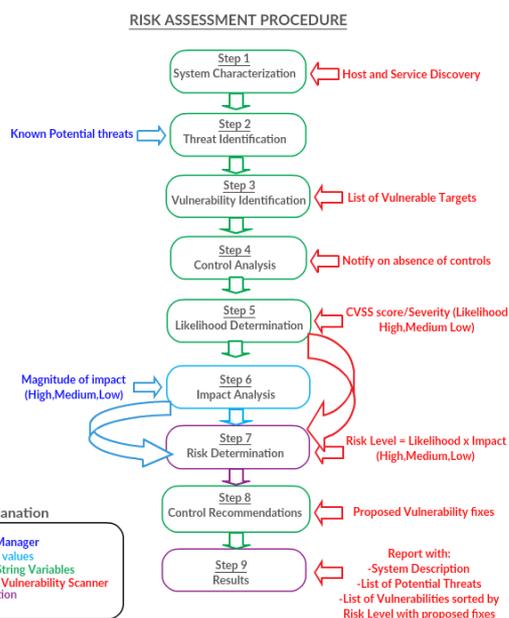


Fig. 5. Risk Assessment with a vulnerability scanner

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.

VIII. CONCLUSION

The main conclusion that can be drawn from this comparison is that all three of these vulnerability scanners can 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 test 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 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.

REFERENCES

- [1] The Government of the Hong Kong Special Administrative Region, "An overview of Vulnerability Scanners", 2008
- [2] Gary Stoneburner, Alice Goguen and Alexis Feringa "Risk Management Guide for Information Technology Systems", Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-30, July 2002
- [3] Hannes Holm, Teodor Sommestad, Jonas Almroth, Mats Persson, (2011) "A quantitative evaluation of vulnerability scanning", Information Management & Computer Security, Vol. 19 Issue: 4, pp.231-247, <https://doi.org/10.1108/09685221111173058>, 2011
- [4] Al-Ayed, A., Furnell, S.M., Zhao, D. and Dowland, P.S. (2005), "An automated framework for managing security vulnerabilities", Information Management & Computer Security, Vol. 13 No. 2, pp. 156-66.
- [5] Andress, M. (2004), "Network vulnerability assessment management", Network World, November 8, 2004