Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
202017 | LIU-IDA/LITH-EX-A--2017/072--SE

Evaluation of two vulnerability scanners accuracy and consistency in a cyber range

Utvärdering av två sårbarhetsscanners med avseende på träffsäkerhet och konsekvens i en cyber range

Erik Hyllienmark

Supervisor : Chih-Yuan Lin
Examiner : Kristian Sandahl

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

One challenge when conducting exercises in a cyber range is to know what applications and vulnerabilities are present on deployed computers. In this paper, the reliability of application- and vulnerability reporting by two vulnerability scanners, OpenVas and Nexpose, have been evaluated based on their accuracy and consistency. Followed by an experiment, the configurations on two virtual computers were varied in order to identify where each scanner gathers information. Accuracy was evaluated with the f1-score, which combines the precision and recall metric into a single number. Precision and recall values were calculated by comparing installed applications and vulnerabilities on virtual computers with the scanning reports. Consistency was evaluated by quantifying how similar the reporting of applications and vulnerabilities between multiple vulnerability scans were into a number between 0 and 1. The vulnerabilities reported by both scanners were also combined with their union and intersection to increase the accuracy. The evaluation reveal that neither Nexpose or OpenVas accurately and consistently report installed applications and vulnerabilities. Nexpose reported vulnerabilities better than OpenVas with an accuracy of 0.78. Nexpose also reported applications more accurately with an accuracy of 0.96. None of the scanners reported both applications and vulnerabilities consistently over three vulnerability scans. By taking the union of the reported vulnerabilities by both scanners, the accuracy increased by 8 percent compared with the accuracy of Nexpose alone. However, our conclusion is that the scanners' reporting does not perform well enough to be used for a reliable inventory of applications and vulnerabilities in a cyber range.

Acknowledgments

I would like to thank my supervisor, Teodor Sommestad, at the Swedish Research Defence Agency. Teodor has continuously given me clear feedback and brought new ideas into this thesis work. I am grateful to my examiner Kristian Sanddahl and supervisor Chih-Yuan Lin for supporting me throughout this process.

Contents

List of Figures

List of Tables

1 Introduction

A cyber range consists of hardware, software, support tools and other capabilities which creates a good environment to arrange events in cyber space [8]. Real world scenarios can be created by forming networks with different configurations. Virtualization is often used to increase the scalability of events in cyber ranges [3], which makes it possible to deploy multiple virtual computers on a single server. Cyber ranges are valuable assets to practice, conduct research and do testing to improve defense against cyber-attacks [8]. Cyber ranges are often used in the military, but also in academia and the private sector [10].

It is desirable for system administrators to do reliable inventory of cyber ranges during events. A reliably inventory of a cyber range includes identifying what operating systems are deployed on the computers, how the computers are configured and if the computers contain vulnerabilities. The inventory is helpful before an exercise to validate if the cyber range has been correctly set up. During an exercise it is desirable to know if configurations on computers have changed. Computer inventory management is a difficult task and the need for automation increase as the ranges becomes bigger [3]. Vulnerability scanners can be used to automatically gather configurations and vulnerabilities on virtual computers. However, vulnerability scanners often produce erroneous results [2].

The vulnerability scanning tools OpenVas and Nexpose has been evaluated and studied in this paper. The evaluation is based on the metrics accuracy and consistency. The evaluation is followed by an experiment, which was made to better understand how each scanner gather information.

1.1 Cyber range event

Two teams usually play against each other in a cyber range event. The "blue" team is a group of defenders and is supposed to detect and protect against cyber-attacks. The "red" team consists of attackers and they perform real world attacks with the objective to gain access to other computers and perform malicious activities. During an event, a "Range Support" team typically operates the support tools. The "white" team is responsible for administrating and monitoring an event. [8]

Events in a cyber range are held contained and isolated from the outside world [10]. The major benefit of the isolation is that activities in the cyber range does not pose a risk of damaging other computer systems. Another advantage of the isolation is that almost all parameters in a cyber range can be fully controlled. This way, sequences from events can be reconstructed and studied at a later point in time.

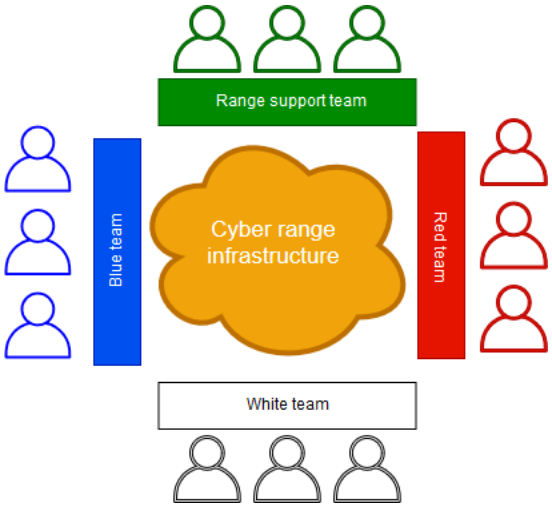

Figure 1.1: Cyber range event

In 2019, a cyber security exercise "Locked Shields" [18] was held, where more than 30 countries and 1200 participants participated. Real world scenarios were constructed, and security incidents could be practiced in a realistic way. The event was carried out in a cyber range, in an isolated environment under safe and controlled conditions. A cyber range is indeed a good place to perform testing, conduct research and for training.

An event consists of four phases. A plan and design for the event is made in the first phase, which is called the "planning phase". The second phase is called the "deployment phase". Computers, networks, applications and services are configured and validated in the deployment phase. The main phase of an event is called the "execution phase", where the event is executed. The execution phase includes simulation of network- and user traffic, data collection and event monitoring. The last phase is called "post-execution" phase. In the post-execution phase, data is analyzed, and the cyber range is sanitized for new events to take place. [8]

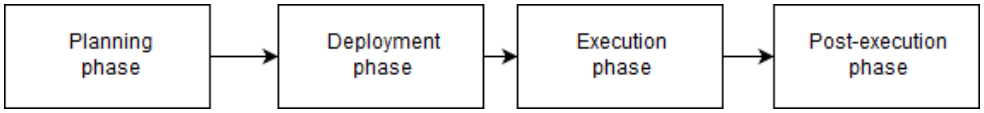

Figure 1.2: Cyber range event phases

1.2 Motivation

Reliable inventory of the cyber range is desirable during the deployment and execution phase of an event. In the deployment phase, tools are often used to automatically configure each virtual machine [3]. Sometimes the deployment of virtual computers fails, and misconfigured virtual computers are deployed in the cyber range. A reliable inventory of each virtual machine gives administrators confidence that the cyber range has been set up as planned.

During the execution phase, configurations may be changed often. Event participants may change system configurations, install new applications, and shut down important parts of the system without noticing the administrators. These changes may cause new vulnerabilities to be introduced in the range. The administrators need to know how the computers are configured in order to execute events as planned. A reliable inventory of the cyber range can discover configuration errors and vulnerabilities, which may or may not need to be fixed. A reliable inventory of the cyber range can provide administrators with the current computer configurations and vulnerability informa-

tion. This data can also be used to visualize computers in the cyber range and their current state [3].

1.3 Research questions

It is important that a vulnerability scanner provide complete and precise results. Complete results mean that the results provides a full picture of the cyber range, without excluding any information. Precise results mean that the results do not contain errors. Erroneous results may lead an administrator to take incorrect decisions. If the results do not provide a complete picture of the cyber range, good decisions may not be taken, because the basis for a decision was not there. Our first research question is:

RQ 1: How accurate are the tools?

Moreover, a scanning tool for reliable inventory should consistently report the same results over multiple scans. In consistent results, incorrect result samples origins from the very same error(s). A tool might incorrectly report on one vulnerability in the first test, but, in the second test, the tool does not report on the vulnerability, it instead includes another incorrect vulnerability. In this case, the accuracy for the two scans are the same but the results would not be consistent. Our second research question is:

RQ 2: How reliable are the tools?

One tool may not be able to perform a reliable inventory on its own. If several tools are combined, the tools may be able to solve different tasks, or one task together, in a better way. A combination of multiple tools may produce better results. Thus, our final research question is:

RQ 3: Can tools be combined to get a better result?

1.4 Delimitations

This thesis will study the inventory of applications and vulnerabilities in cyber ranges with a focus on vulnerabilities registered in the National Vulnerable Database [22]. Two vulnerability scanners are considered for this purpose. To perform better, each scanner is given access to an administrative account on each computer and no firewalls are used. The scanners are evaluated in a single cyber range provided by The Swedish Defense Research Agency [27].

2 Background

This chapter presents a short introduction to vulnerability scanning, application and vulnerability naming schemes, the CRATE cyber range and requirements on tools to use in the evaluation.

2.1 Vulnerability scanning

A vulnerability scanner is a software tool which can be used to automatically find vulnerabilities on a computer. A "Network based" vulnerability scanner is installed on a computer and scans remote computers [2]. A scan is done by sending a specially crafted request to a remote computer, and then comparing the response to vulnerability signatures. Network based vulnerability scanners can perform authenticated scans to improve the accuracy and detect more vulnerabilities on a computer [16]. In authenticated scans, the scanner logs on to a remote computer to gain more access to the computers' resources, such as the file system and the Windows registry [1]. Moreover, the operating systems interface can be queried to access computer information. Depending on the type of account used to log on to the computer, the scanner may or may not access certain resources.

2.2 Reporting standards

The National Vulnerable Database [22] contains dictionaries with applications and vulnerabilities described in standardized formats. The database can be used to find vulnerabilities associated to a software product or to find software products affected by a vulnerability. The Common Vulnerabilities and Exposure (CVE) dictionary contain vulnerabilities indexed with CVE codes. A CVE code is a unique identifier for a publicly known vulnerability. A CVE has the following format

CVE-2018-0599

where the middle part indicates which year the CVE code was assigned the vulnerability, or which year vulnerability was made public [2]. The Common Platform Enumeration (CPE) dictionary contains software named according to the CPE naming scheme. A CPE code is a reference to a hardware platform, an application or an operating system. For example, the CPE code for the application "Internet Explorer" with version "6.0" by the vendor "Microsoft" looks like the following.

[1] Hierarchical database with system information
[2] https://cve.mitre.org/about/faqs.htmlyear_portion_of_cve_id

cpe:2.3:a:microsoft:ie:6.0:*:*:*:*:*:*:*

2.3 Cyber Range And Training Environment

Cyber Range And Training Environment (CRATE) [7] is a cyber range developed by the Swedish Defence Research Agency. The range consists of around 800 servers where many hundreds virtual computers can be deployed. Table 2.1 lists a couple of possible scenarios during an event in CRATE.

Desired information	**Phase**
Virtual computers are deployed	Deployment phase
Applications are installed	Deployment phase
Attacks are performed	Execution phase
New applications are installed	Execution phase
New vulnerabilities are introduced	Execution phase
Virtual computer crashes	Execution phase
Virtual computers are reconfigured	Execution phase

Table 2.1: Event incidents

Deployment scripts are used in CRATE to deploy virtual computers with different operating systems, virtual networks and applications in the deployment phase. Vulnerabilities are introduced on each computer as a result of the deployed applications and configurations. Moreover, CRATE is not always stable, and the scripts may fail. Therefore, it is necessary for the white team [3] to inventory the cyber range in the end of the deployment phase. The inventory in the deployment phase can be compared to future inventories to detect changes on virtual computers.

Many things can happen during the execution phase of events. Virtual computers can crash, attacks are performed and virtual computers are reconfigured. For example, new applications can be installed, services can be disabled, and networking configurations may be changed. It is important for the white team to identify if a computer is no longer available as soon as possible. The current application and vulnerability information is needed for the range support crew to operate their tools and to manage the event. Moreover, the white team desires the current application and vulnerability information to monitor the event.

2.4 Tools requirements

The Swedish Defence Research Agency laid out the following requirements on the tools to be used during events in CRATE.

1. No tool must be installed on the virtual machines.
2. Each tool must scan both Windows and Linux systems.
3. It must be possible to automate each tool.
4. Each tool must report in a recognized and standardized format.

Additionally, free tools were preferred. The following conditions were also defined: 1) Secure Shell (SSH) or Server Message Block (SMB) protocol was enabled on each computer for authenticated scans, and 2) credentials to a privileged account were provided to each computer.

[3] Event administrators

3 Related work

Our study is concerned of inventorying a cyber range with vulnerability scanners. Therefor, this chapter first present studies on vulnerability scanners. Then, a couple of studies which presents alternative methods to validating a computer environment are presented. The distinction between our work and the related work, presented in this chapter, is described in the last section of this chapter.

3.1 Vulnerability scanners

Fonseca et al. [12] questions to what degree web vulnerability scanning results can be trusted. The authors states that it is hard to determine the relevance of scanning results without knowing the rate of false positives and the coverage of the scanner. Three web vulnerability scanners were tested by automatically injecting common types of faults in the source code of applications and then scanning for vulnerabilities. Efficiency was evaluated based on the number of false positives detected. Coverage was measured by manually searching for vulnerabilities in the application and comparing the findings with the scanning results. The authors conclude that web vulnerability scanners produce many false positives and many vulnerabilities were not detected.

W. Qianqian and L. Xiangjun [24] extended a web vulnerability scanner to test against one attack vector. The authors point out that scanners do report false positives. The scan results must therefore be manually verified to detect erroneous results. The authors verified vulnerabilities on a remote computer, performed a scan and confirmed the vulnerability in the scanning results. The author states that the accuracy between web vulnerability scanners differs significantly because of different implementation differences between scanners.

Lukanta et al. [19] extended a web vulnerability scanner with a plugin. Performance of the tool was measured by counting the number of false positives, false negatives, true positives and true negatives. Each vulnerability is classified by exploitation.

Daud et al. [9] evaluated three vulnerability scanners based on the number of vulnerabilities reported when scanning two projects. However, the vulnerabilities were not verified. The author concludes that the scanners detect different kind of vulnerabilities because of their different purpose, i.e. web vulnerability scanning or network vulnerability scanning. Another experiment compared the number of vulnerabilities reported when scanning at two different points in time. The scan results differed because the plugins in the scanner changes over time.

Gordin et al. [14] used OpenVas, Nessus and Metasploit to assess the security of a cloud architecture. The performance of each scanner was evaluated based on the number of open ports detected. The author concludes that Nessus provides detailed information of each detected ports, OpenVas provides good results for being a free tool and Metasploit detected the greatest number of open ports.

Holm et al. [17] evaluated 7 commercial vulnerability scanners based on their accuracy to identify vulnerabilities. Accuracy was measured by counting the number of false positives and false negatives. False negatives were evaluated by randomizing 50 vulnerabilities in a pool which contained all possible vulnerabilities. False positives were assessed by randomizing 40 vulnerabilities from the scanning results, and then confirming each vulnerabilities' presence on the computer. Both authenticated and unauthenticated scans were done in a network of 28 virtual computers. The authors conclude that vulnerability scanners are helpful, but do not report accurately. The scanners report around half of all vulnerabilities present in the network, and the scanning result do contain false alarms. The authors conclude that the accuracy increases by doing authenticated scans. Finally, the authors combined the vulnerabilities reported by all scanners, which increased the detection rate with a low false positive rate.

3.2 Environment validation

Doelitzscher et al. [11] deploys agents on each computer in a cloud infrastructure to validate its security status. The cloud infrastructure consists of virtual machines, which are subjected to frequent changes. An agent is programmed in Python and is programmed to collect a specific piece of information. The agent collects information and sends it to a central server which provides administrators with an overview of the cloud. The authors conclude that using agents to validate a cloud infrastructure is beneficial because of its light weight and flexibility.

Al-Matari et al . [20] utilizes cyber security tools to audit cloud networks. A comparison of tools which can be used for gathering information, scan and exploit a computer is presented. The authors conclude that a single tool cannot perform a complete audit but tools should be combined to solve different tasks. Furthermore, most tools require training to operate and most tools are commercial. A future work includes building a framework which integrates the various tools controlled via a user interface.

Russo et al. [25] propose a framework to automate the design, deployment and validation of an event in a cyber range. The authors proposes an extension of the TOSCA standard called Scenario Definition Language (SDL). SDL allows the infrastructure (applications, files, policies etc) to be written as code (Infrastructure-as-Code) and to model the infrastructures' components. SDL is used to create a declarative specification for the design of the event and each SDL type includes a piece of test code which can be used for validation after deployment. The design is then translated into a specification written in the declarative programming language Datalog. Then, Datalog clauses are run in an engine, which produces queries to run against conditions (often a functional requirement) which must not be violated. After the deployment of the event, scripts in the Datalog clauses are extracted and executed to validate the deployed infrastructure against the clause. Something in the deployment went wrong if the test script failed.

3.3 Our work

In this study, vulnerability scanners are used for the purpose of an inventory. We quantify the consistency and accuracy of vulnerability scanners to inventory not only vulnerabilities, but also applications. We also explore how a scanner works by identifying where the scanner collects its information. Unlike other studies, our evaluation combines true positives, false positives and false negatives to quantify accuracy (f1-score) into a single number. Also, consistency between vulnerability scans is quantified into a single number. Moreover, our evaluation was done in a cyber range with 35 virtual computers, which makes the evaluation larger than other studies.

4 Tools

This chapter first introduce two vulnerability scanners which satisfies the requirements presented in section 2.4. Then, a couple of excluded vulnerability scanners are listed.

4.1 OpenVas

The Greenbone Vulnerability Management [15] system consists of four main components, illustrated in in Figure 4.1 and described in the list below.

1. The OpenVas scanner: scan engine used to execute Network Vulnerability Tests
2. The Greenbone Community Feed: feed with more than 50 000 open source NVTs.
3. The Greenbone Vulnerability Manager: handles communication and management between components and clients
4. The Greenbone Security Assistant: web interface to control the scanner

The scanner can also be controlled and automated [1] via the "Greenbone Management protocol (GMP)", which is an XML based protocol used to communicate with the GVM.

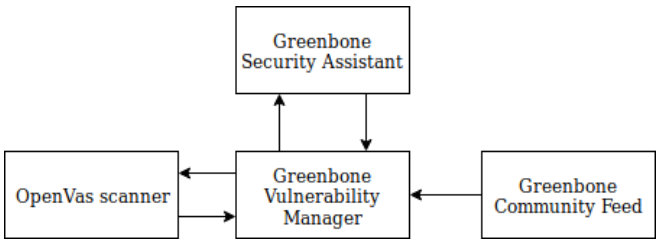

Figure 4.1: OpenVas architecture

4.2 Nexpose Community

The main component of the Nexpose Community Edition scanner [13] is the Security Console, as illustrated in Figure 4.2. The Security console is an application which handles communication between other components. It can be used to configure sites, configure credentials, generate reports and display data. A site consists of assets, a scan template, a scan engine and other settings. An asset is a device with an IP address. The security console can be operated via a web interface or through a HTTPS [1] based API.

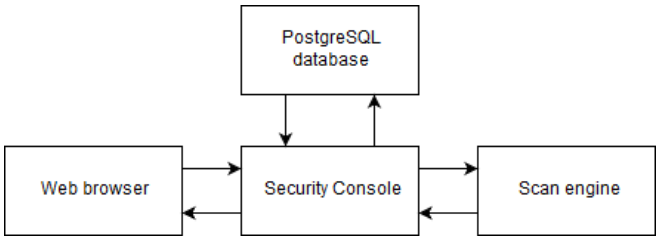

Figure 4.2: Nexpose architecture

4.3 Other scanners

Scanner were searched for in other research papers, on the web, and in technical reports. Many different vulnerability scanners with various purposes and limitations exists. A few discarded scanners, with their limitations, are listed in Table 4.1. Scanners were discarded because they violated the requirements presented in section 2.4 ("Vuls" for example), because they have limited use ("sshdefaultscan" and "CVE-scan" for example) or because they were not free of use. OpenVas and Nexpose provided the broadest functionality without violating the requirements which made them a good choice for our evaluation.

Tool	**Limitation(s)**
Vuls [2]	Scans Linux only
Vulscan [3]	Does not support authenticated scans
Nessus Home [4]	Strictly for home use
CVE-scan [5]	Does not support authenticated scans
Open-AudIT	Identifies applications only Does not work on all Linux computers
sshdefaultscan 0.4.0 [6]	Detects only SSH vulnerabilities
OWASP dependency check [7] [8]	Scans dependencies only
Qualys FreeScan [9]	Requires internet access

Table 4.1: Discarded scanners

[1] Hypertext Transfer Protocol Secure

5 Method

This chapter presents a quantitative evaluation on the scanners' accuracy and consistency and an experiment which was meant to identify where on a remote computer each scanner finds its information. Configurations were varied between two vulnerability scans in the experiment. The experimental setup along with the configurations, which were varied in the experiment, is presented in the first section of this chapter. Then, a vulnerability definition, techniques used to identify applications/vulnerabilities in the network, and the methodology used to classify applications/vulnerabilities as present or absent on a computer is presented. The accuracy and consistency metrics are described in detail in the last section of this chapter.

5.1 Experimental setup

The quantitative evaluation was done in a typical organization, illustrated in Figure 5.1, used in events held in CRATE. The organization contained 35 virtual computers and consisted of three virtual networks connected with a router. Each virtual computer had a Windows or Linux operating system with different applications and vulnerabilities.

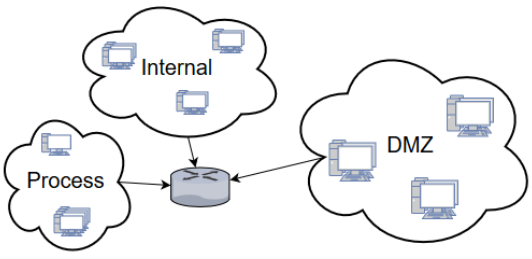

Figure 5.1: Organization

Two computers, with operating systems Windows 7 and Linux Ubuntu 16.04, were chosen from the organization to be used in the experiment.

Varying services

A service applications usually run in the background and provides services to other applications [5]. User applications often require multiple services and removing a service can cause an application to crash. The Windows Service Manager was used to configure services. The following two tests were done on the Windows 7 computer.

1. Disable services in the Service Manager (Services.msc)
2. Remove sub keys in the "Services" registry

Varying Dynamic Link Library

Dynamically Linked Libraries (DLLs) are libraries used by applications to solve common tasks on Windows computers. A system DLL enables applications in user mode to call processes running in Kernel mode [5], as illustrated in Figure 5.2. Windows has a default registry key, which hold paths to DLL files. In this test, sub keys in the "Known DLLs" registry key [1] on the Windows computer were removed.

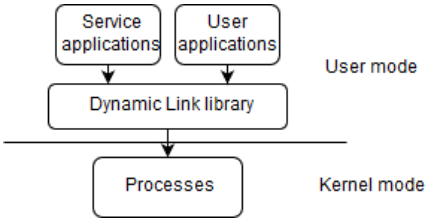

Figure 5.2: Applications, Dynamic Link Memory, Kernel mode and User mode

Varying user applications

When an application is installed executable files and configuration files are copied to the file system. The Windows registry is also configured on Windows computers. The recommended way to install applications on Windows is to use the "Windows Installer", which takes care of placing files in the file system and configuring the Windows registry [5]. In the "App paths" registry key, information is stored of registered Windows applications installed with the Windows Installer. On Linux, the package manager is the preferred way to install applications, which usually copies files to the "/usr/bin" directory when applications are installed [23]. The following tests were done.

1. Uninstall applications in the Control Panel (Windows).
2. Remove sub directories in the "/usr/bin" directory (Linux).
3. Remove sub directories in the "/usr/sbin" directory (Linux).
4. Remove sub keys in the "App paths" registry key (Windows)

Applications can also be installed in a non-standard way. For example, with a custom installer in Windows. By using a non-standard installation method, files can end up in various places and the Windows Registry can be arbitrarily configured. The following tests were done on the Windows 7 computer.

1. Remove sub keys in the "Uninstall" registry key
2. Move sub directories in the "Program Files" directory to another location (Windows).

[1] HKLM\SYSTEM \CurrentControlSet \Control \Session Manager \KnownDLLs

5.2 Definitions and techniques

It is necessary to know which applications and vulnerabilities are really present on a computer to evaluate accuracy. Classifying an application/vulnerability as presence or absence on a computer is a binary classification problem. The *ground truth*, refers to applications and vulnerabilities really present on a computer. This section first defines a vulnerability. Then, techniques used to identify installed applications, application versions and configurations on computers are described. This information is required to determine if an application is installed on a computer, and to determine if a computer was vulnerable. All techniques are listed in Table 5.1 and described in this section.

Verify	**Technique**
Application name	Control Panel
	Start menu
	Package manager
	WMI command-line
Application version	Windows Registry
	Check file system
	Start application

Table 5.1: Vulnerability verdict techniques

Vulnerability definition

The National Vulnerable Database was used to find vulnerability information. The NVD presents each information for each vulnerability in multiple sections. A computer was determined vulnerable if the following two criteria were satisfied.

1. The required application(s), according to the section "Known Affected Software Configurations" in the NVD, was installed.
2. The computer was configured according to the description in the "Current Description" section in the NVD.

In some cases, the required application(s) was described in the "Current Description" section and not the "Known Affected Software Configurations" section in the NVD.

Windows Start Menu

Windows computers has a Start Menu, which lists installed applications, in the lower right corner of the screen. Applications could be identified by clicking on the Start menu and search through the menu of installed applications. [5]

Windows Control Panel

The "Applications and Features" application in the Windows Control Panel provides human readable list of applications, and their versions, installed with the Windows Installer. The list was accessed by clicking on the Control Panel icon in the Windows Start Menu followed by a double click on the "Programs" icon and a double click on the "Applications and Features" icon. If Applications and Features listed an application, the application was considered installed on the computer. [5]

Linux Package manger

Each Linux distribution has a package manager, which can be used to list installed applications and their version numbers. The list was accessed by opening a terminal prompt, followed by the package managers' command to list installed applications. The "grep" command was useful to

search through the list of installed applications. If the package manager listed an application, the application was considered installed on the computer. [23]

WMI command-line

The "WMI command-line (WMIC)" [2] is a tool which provides an interface to interact directly with Windows Management Instrumentation (WMI) classes [3]. We accessed the tool by opening the Windows Command Prompt and executed the "wmic" command. Once the WMIC tool had started, a command was issued to list installed applications on the computer. The tool lists the applications' name and version. If WMIC listed an application, the application was considered installed on the computer.

Windows Registry

The Windows Registry was queried to identify configurations and application versions. A string search (ctrl+f) was done with an applications' name, version or a specific file name.

Launching applications

The application could be launched to identify application version or configurations. On Windows, an application could be started by double clicking on the executable file, or by clicking on the applications' icon in the Windows Start Menu. On Linux, an application could be launched from the terminal. To find the version number, the applications could be started with the "-v" flag in the Linux terminal. On Windows, the applications' version could be found in the "Home" or "About" tab of the application. In few cases, further interaction with the application was made to find configurations. For example, the python program was launched, and a module imported to confirm the modules' presence. If an application could be launched, the application was considered installed.

File search

The file system was searched to find application versions and configurations. Important files were "readme" files, "about" files, "install" files, DLL files, or the executable file to an application. On Linux, the "whereis", "locate" and "apt-cache" terminal commands were helpful. On Windows, file search was done with the search bar in the File Explorer.

5.3 Classification methodology

It was necessary to determine if applications and vulnerabilities were present on computers, i.e. if they were part of the ground truth, to quantify accuracy. True positives are applications/vulnerabilities correctly labeled, by the scanner, as part of the ground truth. False positives are incorrectly labeled as part of the ground truth. False negatives are applications or vulnerabilities, which belongs to the ground truth, but which the scanner fails to label as part of the ground truth. True negatives are samples correctly labeled as not part of the ground truth. This section describes how it was determined to which set a reported application and vulnerability belongs to. [6]

Collect "ground truth" applications

The following process was done before the scanning to collect applications: 1) randomize one computer in the organization, 2) identify installed applications, and 3) randomize one application. The process is described below and illustrated in Figure 5.3.

[2] https://docs.microsoft.com/en-us/windows/desktop/wmisdk/wmic
[3] WMI classes uses the Windows API to access data or to the control operating system.

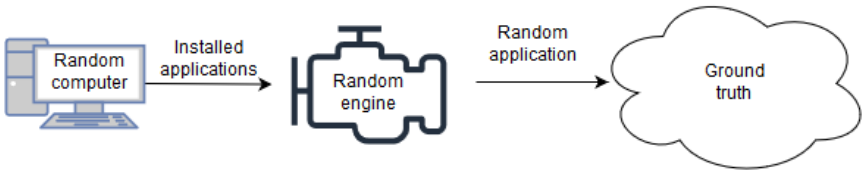

Figure 5.3: Collect ground truth, applications

1. A list of all computers was made. A number between 0 and the list length was randomized with the Python Random module [28]. The random number pointed out the index of which computer to pick in the list.
2. To find installed applications, the "Programs and Features" in the Windows Control Panel was used. On Linux, the package manager was used to retrieve a list of installed applications.
3. The length of the list with installed applications was counted. A number between 0 and the list length was randomized. The random number pointed out the index of which application to pick in the list.

The process was repeated until 50 applications from the ground truth had been identified, which should be enough to represent a variety of applications on different platforms.

Collect "ground truth" vulnerabilities

The process to collect the ground truth for vulnerabilities, before the scanning, was similar to the process for collecting applications. The process is illustrated in Figure 5.4.

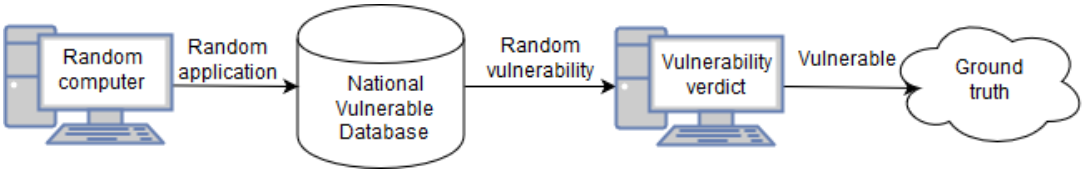

Figure 5.4: Collect ground truth, vulnerabilities

First, a computer and application were randomized in the same way as already described. Then, the chosen application was used to find a vulnerability. A "Non-exact keyword search" for vulnerabilities was made in the NVD database [4] with the application as search parameter. The search input was varied in the following way until at least 1 vulnerability for the given application was retrieved in the search.

1. Full application name and full application version number
2. Full application name and shortened version number
3. Shortened application name and full application version number
4. Shortened application name and shortened application version number
5. Full application name and remove version number
6. Shortened application name and remove version number

[4] https://nvd.nist.gov/search

Then, a number between 0 and the number of retrieved vulnerabilities (CVE codes) in the search results was randomized. The random number was used as the index of the vulnerability to pick in the list. A new vulnerability was randomized if the vulnerability had already been inspected on the same computer. If no vulnerability was found the whole process was restarted. The vulnerability was added to the ground truth if the computer was determined vulnerable. The process was repeated until 50 vulnerabilities had been identified.

Confirm "ground truth"

The ground truth and the scan results were compared, as illustrated in Figure 5.5, after the organization had been scanned. The aim was to determine if the scanner reported on the ground truth or not.

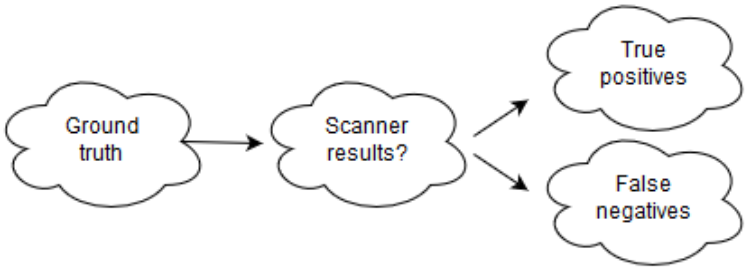

Figure 5.5: Confirm ground truth

This was done by going through each ground truth sample, collected before the scan, and check if the report produced by the vulnerability scanner contained the application or vulnerability. If the ground truth sample had indeed been reported, a true positive had been identified. If the ground truth sample had not been reported by the tool, a false negative had been identified.

Verify scanner results

After the organization had been scanned, 50 applications and 50 vulnerabilities from the results of each scanner were verified according to the process illustrated in Figure 5.6. The following process was done: 1) randomize computer, 2) randomize result sample, and 3) verify result sample.

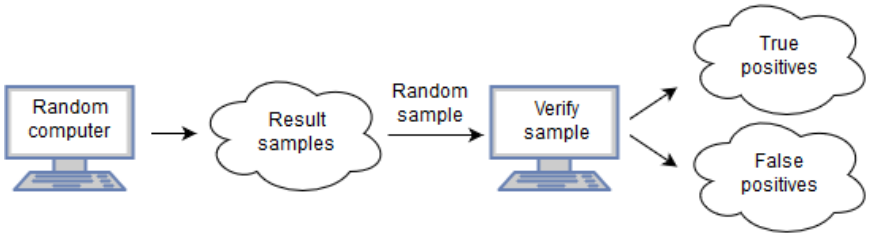

Figure 5.6: Verify scan result samples

All reported applications and vulnerabilities were collected in two different lists. To randomize a sample for one of the lists, the python Random module was used to take a random number within the list length which pointed out the index to a result sample to verify.

The verification of the applications and vulnerabilities were done with the techniques listed in Table 5.1 and described in section 5.2. A sample which was verified as part of the ground truth was regarded as a true positive. Otherwise, the result sample was counted as a false positive.

Combine scanner results

The vulnerabilities reported by both scanners were combined with their union and intersection to calculate the scanners' combined accuracy. The same ground truth samples (50 vulnerabilities) and verified vulnerabilities (100 vulnerabilities) previously identified were reused.

Vulnerabilities reported by both vulnerability scanners were considered to calculate the union. This means that either one of the scanners had to report on a ground truth vulnerability sample for the sample to be regarded as a true positive. To verify scanner results, the true positives and false positives from all verified vulnerabilities (100 vulnerabilities) from both scanners were simply added together (no duplicates existed).

Only vulnerabilities reported by both scanners were considered to calculate the intersection. To confirm a ground truth vulnerability sample, both scanners had to report on the vulnerability in order for the vulnerability to be counted as a true positive. A ground truth sample must have not been reported by any of the scanners to be regarded as a false negative. In the verification process, vulnerabilities which were falsely reported by both scanners were regarded as false positives. Vulnerabilities, which were correctly reported by both scanners were regarded as true positives.

5.4 Metrics

This section describes how the metrics accuracy and consistency were quantified. First, the metrics precision and recall are described, which were used to evaluate accuracy.

Precision and recall

Precision and recall were originally used in the field of information retrieval to measure the performance of an algorithm. Precision, given by formula 5.1, describes the "purity" of the retrieval, which is the rate of correctly labeled applications/vulnerabilities. Recall, given by formula 5.2, quantifies the completeness of the vulnerability reporting. [4]

$$\texttt{Precision} = \frac{\texttt{true positives}}{\texttt{true positives} + \texttt{false positives}} \tag{5.1}$$

$$\texttt{Recall} = \frac{\texttt{true positives}}{\texttt{true positives} + \texttt{false negatives}} \tag{5.2}$$

Accuracy

Precision and recall should be combined to give a complete and precise (informally called accuracy [6]) measure. The f_1-score calculates accuracy by taking the harmonic mean of precision and recall as in equation 5.3. Precision and recall is often inversely related [4] and the f1-score weights the two evenly.

$$f_1 = 2 \cdot \frac{\texttt{precision} \cdot \texttt{recall}}{\texttt{precision} + \texttt{recall}} \tag{5.3}$$

Consistency

To calculate consistency, the number of deviations between vulnerability scans were first counted. A deviation is an application or a vulnerability which was reported in some of the scans, but not in all scans. Then, the samples (applications/vulnerabilities) reported in all the scans were counted. Consistency was finally quantified with the formula

$$\texttt{Consistency} = \frac{samples_1 \cap .. \cap samples_n}{(samples_1 \cap .. \cap samples_n) + deviations} \tag{5.4}$$

where n is scan number n. To achieve full consistency, each scan had to report the exact same applications and vulnerabilities over all vulnerability scans.

6 Results

This chapter first presents the accuracy and consistency results from the quantitative evaluation. The data from the experiment, where consistency between scans was observed as computer configurations were changed, is then presented. The complete data is listed in the appendix.

6.1 Accuracy

The accuracy for applications and vulnerabilities are listed in Table 6.1. The table presents the accuracy for each scanner individually, and for the scanners combined with the union and intersection.

Scanner	Attribute	FP	FN	TP	Precision	Recall	Accuracy
Openvas	Applications	4	35	61	0.94	0.64	0.76
	Vulnerabilities	13	27	60	0.82	0.69	0.75
Nexpose	Applications	1	6	93	0.99	0.94	0.96
	Vulnerabilities	7	29	64	0.9	0.69	0.78
Union	Vulnerabilities	20	21	109	0.84	0.84	0.84
Intersection	Vulnerabilities	14	35	76	0.84	0.68	0.76

Table 6.1: Accuracy (abbreviations: false positives (FP), false negatives (FN) and true positives (TP))

Nexpose performed 26 percent (20 percent points) better than OpenVas with regards to applications and 4 percent (3 percent points) better with regards to vulnerabilities. The scanners, combined with the union, performed 8 percent (6 percent points) better than Nexpose alone and 12 percent (9 percent points) better than OpenVas alone. When combined with the intersection the accuracy was 1 percent better than for OpenVas alone and 3 percent worse compared to Nexpose accuracy alone.

6.2 Consistency

Table 6.2 presents the total number of unique applications and vulnerabilities and the total number of deviations identified during three vulnerability scans. Table 6.3 presents the consistency results for OpenVas and Nexpose over the three scans.

	Unique		Deviations	
Scanner	applications	Vulnerabilities	applications	Vulnerabilities
OpenVas	597	81829	25	0
Nexpose	13669	93888	0	35

Table 6.2: Number of reported applications and vulnerabilities

Scanner	Applications	Vulnerabilities
OpenVas	0.9598	1
Nexpose	1	0.9996

Table 6.3: Consistency evaluation

Nexpose reported applications in a consistent way. However, one vulnerability (CVE-1999-0524) was not reported in all three scans. OpenVas reported vulnerabilities in a consistent way, but not applications. 25 applications were reported in one or two of the scans, but not in all three scans.

6.3 Interfering with scanners information gathering

This section presents the results from the experiment, which was done after the quantitative evaluation. A comparison of reported vulnerabilities between OpenVas and Nexpose is illustrated in Figure 6.1.

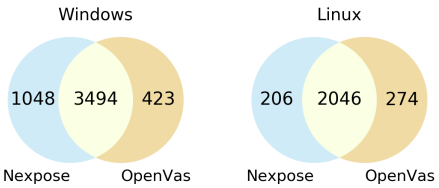

Figure 6.1: Vulnerabilities comparison

Consistent variables

The following variables in experiment yielded consistent results for both OpenVas and Nexpose.

1. Remove sub keys in the "Services" registry
2. Remove sub keys in the "Known DLLs" registry key

Additionally, OpenVas reported consistent results when services were disabled. Nexpose reported consistent results when "/usr/sbin" subdirectories were removed and when the "App paths" registry sub keys were removed.

Inconsistent variables

Inconsistent results are illustrated in figures 6.2 - 6.11. The "Benchmark" circles in figures 6.2 - 6.11 presents the number of applications/vulnerabilities reported in the first scan, when nothing on the computer had changed. The circles on the right side in each Figure presents the number of applications/vulnerabilities reported when the variable was varied.

Combinations

Two tests were done when the following variables were tested in combination.

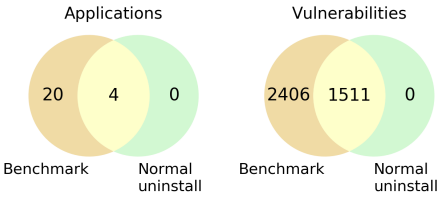

Figure 6.2: OpenVas "Control Panel uninstall"

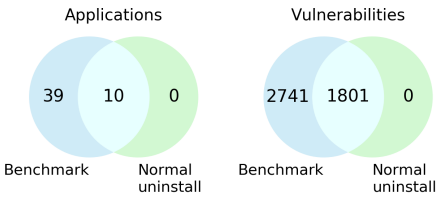

Figure 6.3: Nexpose "Control Panel uninstall"

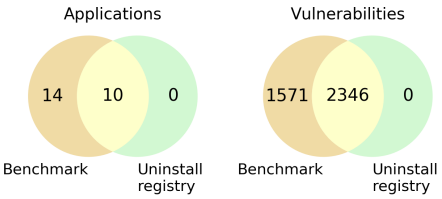

Figure 6.4: OpenVas "uninstall registry"

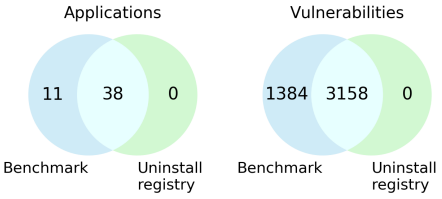

Figure 6.5: Nexpose "uninstall registry"

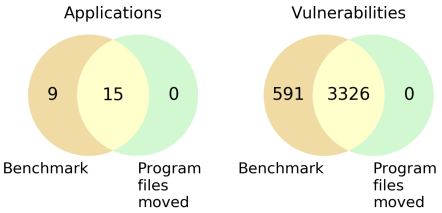

Figure 6.6: OpenVas "program files moved"

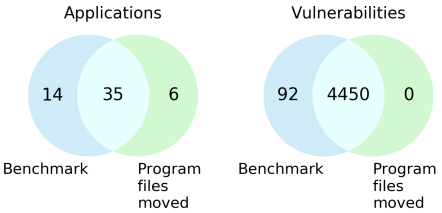

Figure 6.7: Nexpose "program files moved"

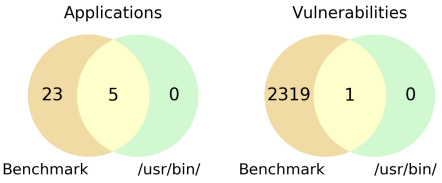

Figure 6.8: OpenVas "/usr/bin"

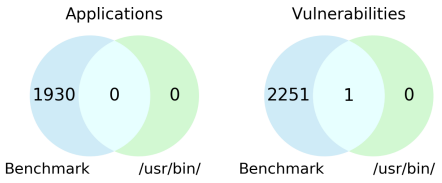

Figure 6.9: Nexpose "/usr/bin"

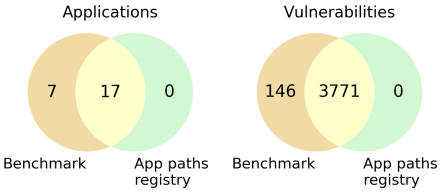

Figure 6.10: OpenVas "App Paths"

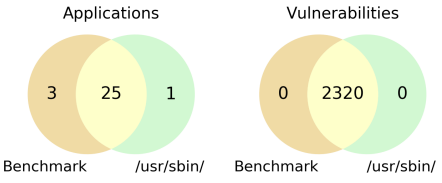

Figure 6.11: OpenVas "/usr/sbin"

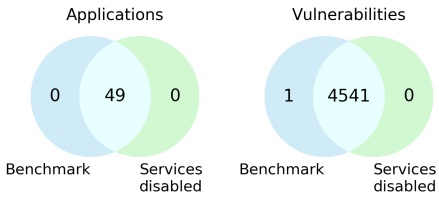

Figure 6.12: Nexpose "services disabled"

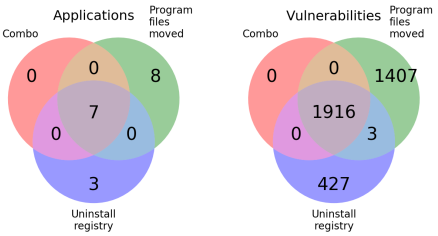

Figure 6.13: Combination 1, OpenVas

1. Subdirectories in the "Program files" moved and subkeys in the "Uninstall" registry removed
2. Subdirectories in the "Program files" moved and subkeys in the "App paths" registry removed

The first combination was tested with both OpenVas and Nexpose. The second combination was only tested with OpenVas. The results are illustrated in figures 6.13, 6.14 and 6.15. The colored circles have the following meaning:

1. Green: applications/vulnerabilities detected when only the first variable was varied
2. Purple: applications/vulnerabilities detected when only the second variable was varied
3. Red: applications/vulnerabilities detected when the first and the second variable were varied simultaneously

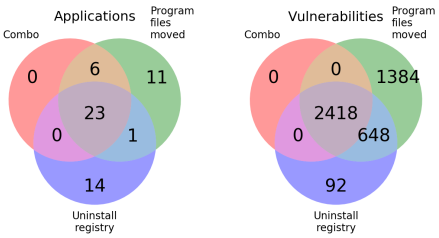

Figure 6.14: Combination 1, Nexpose

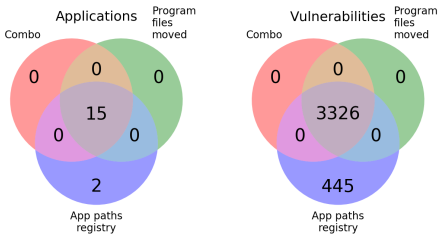

Figure 6.15: Combination 2, OpenVas

7 Discussion

This chapter is divided into two section. The results from the evaluation and the experiment is discussed in the first section. This is followed by a discussion of the methodology used in this study.

7.1 Results

This section discusses the results from the evaluation and experiment. First, the quantitative accuracy- and consistency results from the evaluation are discussed. Then, reporting standards and the scanners' techniques to find information are discussed.

Accuracy

The results show that the accuracy between OpenVas and Nexpose differs. Nexpose performed better than OpenVas, both with regards to applications and vulnerabilities. By taking the union of the vulnerabilities reported by both scanners, the accuracy increased. The scanners combined performed better than each scanner alone. The quite low (0.64) application recall value for OpenVas had a large impact on its accuracy result. Nexpose, that had a high precision value and a rather high recall value for applications, produced a high accuracy value.

Table 6.1 shows how the bad accuracy is more a result of bad recall than bad precision. The precision value was higher than the recall value, for applications and vulnerabilities, for both OpenVas and Nexpose. This result is aligned with the results from Holm et. al [17], who concluded that erroneous scan reporting origins from a high false negative rate rather than a high false positive rate.

CPE naming scheme

Nexpose reported applications in a structured XML format (including vendor, product and version). According to the documentation, Nexpose does adopt to the CPE naming scheme. However, few applications were assigned CPE codes in the web interface, and CPE codes were not available at all in the XML report. OpenVas reported applications only with CPE codes.

An obvious advantage of structured XML reporting is that the recall increase because applications without a CPE code can be reported. A problem with adopting to the CPE naming scheme, is that the scanner must map an applications name and version to a CPE code [21]. The scanner may very well fail to find the CPE code for an application, which further decreases the recall.

The CPE naming scheme was too poorly implemented in OpenVas and Nexpose for it to be useful. OpenVas reported on more applications with CPE codes than Nexpose, yet few applications were reported (low recall). Nexpose reported only on a handful of applications with CPE codes. The CPE standard can be useful to automatically map applications to vulnerabilities [21] if implemented in a good way.

Consistency

Nexpose was not fully consistent because of one CVE code (CVE-1999-0524). The vulnerability was flagged because a computer advertises itself on the network and gives away to which subnet it belongs 1. The vulnerability origins from a network configuration and may have to do with inconsistencies in the network of the cyber range. Nexpose could be regarded as consistent with the assumption that the scanner is deterministic and that the vulnerability was caused by inconsistencies in the network. However, small inconsistencies are problematic and do decrease the reliability of the inventory. OpenVas application reporting was quite inconsistent (0.96) and inaccurate (0.76), which is not reliable when inventorying a cyber range.

Inaccurate and consistent reporting indicate that the reporting contains errors, but that the same errors are repeated. Consistent and inaccurate reporting is more desirable than inconsistent and accurate reporting, because the cause of erroneous reporting can be better understood if the scanner consistently report the same errors [29]. For example, many errors may be caused by a certain application or a certain configuration on the computer. Our evaluation indicates that the scanners report rather consistently, but that the reporting do contain errors, which is better than the other way around.

Another advantage of consistent reporting is that the scanner can be used for change detection. Computer configurations may change when the computer is attacked, crashed services may remove vulnerabilities or an administrator may mistakenly misconfigure a computer. Then, a consistent vulnerability scanner can be used to detect such changes, even if the reporting is not accurate.

Scanner information gathering

The results indicate that OpenVas and Nexpose gather much of their information from the "uninstall" registry (Figure 6.4 and 6.5), "program files" folder (Figure 6.6 and 6.7) and the "/usr/bin" directory (Figure 6.8 and 6.9)). When emptying the "/usr/bin" directory, almost no applications and vulnerabilities were reported by any of the scanners, which indicates that most information is collected in this directory. Additionally, OpenVas gathers information from the "app paths" registry (Figure 6.10), but Nexpose do not. Neither OpenVas or Nexpose care for the "Service" registry, but Nexpose did report one less vulnerability (CVE-2015-0008) when services were disabled. Moreover, we believe that a scanner can be easily tricked by changing the file system and the Windows registry to avoid detection by the red team in a cyber range event.

We noticed that Nexpose is more inclined to adjust results based on the available information. In Figure 6.7, 6 additional applications were reported when moving the "program files" folder. This is a result of an adjustment made by Nexpose. Nexpose could no longer identify the version number of the applications, and therefor only reported the "vendor" and "product" in structured XML format for the 6 applications. In the case with OpenVas, such adjustments were not made, probably because a CPE code could not be identified with lesser information.

Variable combinations

In combination 1 for Nexpose (venn diagram 6.14), 1 application and 648 vulnerabilities were reported when varying the variables separately, but not when both variables were varied. The reason for this is that the scanner perform logical OR checks on multiple configurations to see if the application is present on the computer. The same thing can be seen for 3 vulnerabilities in Figure 6.13 for OpenVas.

For the most part, OpenVas and Nexpose does not check multiple locations to find information. The tests are rather easy and checks for an application in a single place at the time. If this would not be the case, the dark green areas in Figure 6.13 - 6.15 would have larger numbers.

Future work

As a future work, we believe it would be interesting to continue the research for tools which can be used to reliably inventory a cyber range. New tools and techniques are developed rapidly, which could possibly be used for cyber range inventory. Future work can include an inventory of not only applications and vulnerabilities, but also connectivity between computers and registered user accounts on each computer.

7.2 Method

This section discusses problems with the methodology, which may threaten the reproducibility, reliability and validity of the study.

Scanner choice and settings

Different techniques can be used to pick the right scanner for an experiment [19] [24]. We picked scanners based on fixed requirements and searched for tools, which adheres to the requirements, on the Internet and in research papers. However, there are loads of tools for similar tasks and effectively finding a good set of tools to evaluate is not an easy task. Additionally, most tools are commercial [20], which made the process more time consuming, as it is not always clear on the web page and in the tools' documentation if they are free of use. Other tools most certainly exists, which can be used for the purpose of this study and might have given different results.

In this study, authenticated scans with the default (and most common) scan configurations were used. New versions of the knowledge base of each scanner was used to give the each scanner an equal opportunity to perform its best [9]. Furthermore, it was manually confirmed that SSH and SMB worked on each computer before scanning. However, if other scanner settings would have been applied, the outcome may have been different.

Automation

The automation process made it possible to run multiple identical and consecutive tests [26]. The whole organization could be scanned in a time period less than an hour. Configuring a vulnerability scanner requires skills and the process is prone to errors. By automating the tests, mistakes as a result of the tool operators' skills got reduced. We believe the automation process increased the reproducibility and reliability of the study.

Verdict

Several steps in the evaluation required a to make a verdict whether applications listed in the National Vulnerable Database were installed on a virtual computer. Such verdict is influenced by the experimenters' technical skill. An experimenter with a different level of technical skill might have gotten different results. The first reason for this is that less errors would be introduced in the verification step, when applications and vulnerabilities were classified as true positives, false positives and false negatives. Also, an experimenter with better knowledge would have less probability of introducing errors when gathering applications and vulnerabilities on computers before the experiment.

Keyword search

The process, described in section 5.3 of searching for vulnerabilities in the National Vulnerable Database was systematic to a certain degree. The instructions are not too explicit and another experimenter might have done the keyword search with slightly different search terms. Other vulnerabilities may have been included when collecting the ground truth if other search terms had been used. I do not believe the keyword search process impacted the reliability, but the lack of an clear search process weakens the reproducibility of the evaluation.

Population

There was a lack of diversity among operating systems included in the evaluation. The organization contained mostly Windows 7 computers and most Linux computers were of the same distribution. However, different versions of an operating system works similarly. Thus, the lack of diversity in the organization must not be a major flaw. Moreover. a vulnerability is usually tied to an application itself, independently of the operating system. Furthermore, the same applications were often installed on multiple computers. Applications such as Adobe, VLC and Firefox were very common. This fact reduces the validity of the study, as only a small group of applications were considered in the evaluation.

Definitions

The definition of a "vulnerability" was aligned with the vulnerability definition of a vulnerability scanner. That is, a vulnerability scanner detects potential vulnerabilities, but does not exploit vulnerabilities [26] [2]. One way to be sure a vulnerability is present is to exploit it. Exploiting each vulnerability would have increased the validity of the study, but it would also be time consuming and increasingly dependent on the experimenters' technical skills. We believe the approach used in the evaluation was good enough to answer the research questions in a good way.

Only applications advertised by the operating system [1] were considered when collecting the applications used as the "ground truth". Applications can be installed on a computer in different ways and it is not unambiguous to identify all installed applications. This flaw in the definition reduces the validity of the study because only a subset of all applications installed on the computer was considered.

Consistency metric

The consistency metric did not capture if the same error was produced between multiple vulnerability scans. For example, the 35 errors in Nexpose vulnerability reporting were all present in one of the scans, but 10 of the errors were also present in another scan. A third scan did not report of any of the 35 vulnerabilities. Still, the metric only considered 35 errors rather than 45. Therefore, the results may indicate a more consistent reporting compared to if errors between each scan were to be considered.

Sources

We tried to gather information from a mix of sources, including books, technical reports and research papers. We were cautious to draw conclusions from work older than five years as the field of vulnerability scanning changes rapidly. However, methodologies presented in old work can still be viable. Our methodology was mostly developed in cooperation with employees at the Swedish Research Defence Agency. Other studies provided good insights in vulnerability scanning, but most of them applied very simple and flawed evaluation metrics for accuracy.

[1] Retrieved by "applications and features" and the package manager

The work in a wider context

Vulnerability scanners are intended to identify vulnerabilities and can be used for malicious purposes. We believe determined antagonists will develop techniques to achieve their objective regardless of if information, such as this evaluation, is public or not. Thus, it is better to make information available so that organizations can inform themselves, identify vulnerabilities and take action before antagonists do. Newly discovered vulnerabilities are constantly made public. It is necessary to continuously evaluate security tools and report their performance. This way, vulnerabilities can be detected and handled before any damage is done.

8 Conclusion

The results show that Nexpose is a better scanner than OpenVas to reliably inventory vulnerabilities in a cyber range. Nexpose reported more accurate results and had close to fully consistent vulnerability reporting. However, none of the scanners can be called reliable with accuracy values lower of 0.8, which leads to the first conclusion.

- Neither OpenVas or Nexpose can reliably inventory vulnerabilities in a cyber range

The consistent vulnerability reporting can be used for change detection during the execution phase of cyber range event. That is, if different vulnerabilities are reported over multiple scans, computer configurations have probably changed, which may be useful information for event administrators.

Applications can be better inventoried with Nexpose in a reliable way. The results show fully consistent and more accurate reporting by Nexpose than OpenVas. All errors in the reporting hurts the reliability of a scan. Nexpose application reporting contained very few errors. I believe the errors are within the margin of error. That is, the errors may have been a result of mistakes made by me during the evaluation or because Nexpose applies a slightly different definition of an "installed application". The consistent reporting and high accuracy of Nexpose leads to the second conclusion.

- Nexpose can reliably inventory applications in a cyber range
- OpenVas cannot reliably inventory applications in a cyber range

This can be useful during the deployment- and execution phase, to see what applications are installed on a computer. As a first step, Nexpose can be used to verify that the computer has been deployed as planned in the deployment phase. During the execution phase, newly installed and removed applications can be detected.

OpenVas and Nexpose can be combined to inventory a cyber range in a more reliable way. The accuracy of vulnerability reporting improved by taking the union of both scanners. This result can be combined with Nexpose reliable application reporting. This leads us to the third conclusion.

- Vulnerability scanners can be combined to inventory a cyber range in a more reliable way

9 Appendix

9.1 Network

See organization in table 9.1. Required configurations on remote computer to do authenticated scans over SMB:

Network	OS	Amount
Internal	Windows XP	2
	Windows 7	9
	Gentoo	1
	Windows Server 2003	1
	Windows Server 2008	1
	Windows 10	1
DMZ	Windows XP	1
	Windows 7	1
	Gentoo	3
	Windows Server 2008	3
	Ubuntu	3
Process	Ubuntu 14.04	1
	Windows Server 2012	1
	Ubuntu 16.06	3

Table 9.1: CRATE organization

1. Add a "dword" in the "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" registry with name "LocalAccountTokenFilterPolicy" and value "1".
2. Start the "Remote Registry" service
3. Make sure "File and printer sharing service" is on
4. Add a "dword" in the "HKLM\SYSTEM\CurrentControlSet\Control\Lsa registry with name "forceguest" and value "0". (Windows xp only)

9.2 Evaluation

All duplicate applications and vulnerabilities were on different computers.

Collected vulnerabilities (ground truth)

True positives

1. OpenVas (23)

    a) CVE-2017-7768, CVE-2017-7761, CVE-2014-1806, CVE-2018-0796, CVE-2016-2563, CVE-2017-7767, CVE-2009-0511, CVE-2018-10115, CVE-2016-1108, CVE-2009-2982, CVE-2018-11529, CVE-2016-2826, CVE-2016-4116, CVE-2016-4109, CVE-2016-2335, CVE-2016-4108, CVE-2017-7760, CVE-2015-1670, CVE-2018-1061, CVE-2014-1806, CVE-2017-7766, CVE-2016-1102, CVE-2015-2526

2. Nexpose (21)

    a) CVE-2017-7768, CVE-2017-7761, CVE-2014-1806, CVE-2018-0796, CVE-2017-7767, CVE-2009-0511, CVE-2018-10115, CVE-2016-1077, CVE-2009-2982, CVE-2016-2826, CVE-2016-1077, CVE-2016-4093, CVE-2016-4208, CVE-2017-7760, CVE-2018-5996, CVE-2015-1670, CVE-2018-1061, CVE-2014-1806, CVE-2012-0008, CVE-2017-7766, CVE-2015-2526

False Negatives

1. OpenVas (27)

    a) CVE-2018-20681, CVE-2006-4138, CVE-2014-2441, CVE-2006-4138, CVE-2014-2441, CVE-2015-4813, CVE-2006-0745, CVE-2006-3014, CVE-2018-1000035, CVE-2017-1000158, CVE-2010-3493, CVE-2016-1077, CVE-2014-7185, CVE-2006-3014, CVE-2016-4338, CVE-2018-1060, CVE-2018-18064, CVE-2018-5996, CVE-2014-0253, CVE-2018-19666, CVE-2015-6607, CVE-2012-0008, CVE-2008-3012, CVE-2010-1634, CVE-2016-1104

2. Nexpose (29)

    a) CVE-2018-20681, CVE-2006-4138, CVE-2014-2441, CVE-2016-2563, CVE-2015-4813, CVE-2006-0745, CVE-2006-3014, CVE-2018-1000035, CVE-2017-1000158, CVE-2016-1108, CVE-2010-3493, CVE-2014-7185, CVE-2006-3014, CVE-2016-4338, CVE-2018-11529, CVE-2018-1060, CVE-2016-4116, CVE-2016-4109, CVE-2016-2335, CVE-2016-4108, CVE-2010-3190, CVE-2018-18064, CVE-2014-0253, CVE-2018-19666, CVE-2015-6607, CVE-2016-1102, CVE-2008-3012, CVE-2010-1634, CVE-2016-1104

Collected applications (ground truth)

1. True positives

    a) OpenVas (15)

        i. OpenVPN 2.3.6-I601, Adobe Flash Player ActiveX 9.0.280.0, PuTTY release 0.66, Mozilla Firefox 45.0.1 (x86 en-US), Microsoft Office Professional Edition 2003 , Adobe Reader 9.1, libisc-export160/now 1:9.10.3.dfsg.P4-8ubuntu1 amd64, 7-Zip 9.20 (x64 edition), virtual/ssh-0, Adobe Actrobat Reader DC 15.010.20060, Java 8 Update 66 8.0.660.18, Java 7 Update 10 7.0.100, Putty 0.67.0.0, Google Chrome 49.0.2623.112, 7-Zip 9.20

    b) Nexpose (44)

i. Mozilla Maintenance Service 43, OpenVPN 2.3.6-I601, Oracle VM VirtualBox Guest Additions 4.1.18, Mozilla Firefox (3.6.28), libxcb-dri3-0/trusty,bow 1.10-2ubuntu1 amd64, Adobe Flash Player ActiveX 9.0.280.0, virtual/perl-Test-Simple-1.1.2, sys-apps/findutils-4.4.2-r1, uuid-runtime/now 2.20.1-5.1ubuntu20.7 amd 64, Microsoft SQL Server 2012 (64-bit), dev-perl/HTML-Tagset-3.200.0-r1, dpkg-dev/now 1.18.4ubuntu1.1 all, sys-apps/debianutils-4.7, libc-dev-bin/now 2.19-0ubuntu6.6 amd64, PuTTY release 0.66, time/trusty, now 1.7-24 amd64, dev-utils/kbuild-0.1.999820131130-r1, FileZilla Client 3.5.3, Microsoft .NET Framework 4.5.1, libudev1/now 204-5ubuntu20.15 amd64, Microsoft Office Professional Edition 2003 11.0.5614.0, Mozilla Maintenance Service 45.0.1, Mozilla Firefox 45.0.1 (x86 en-US), mail-mta/nullmailer-1.13-r5, Adobe Reader 9.1, libisc-export160/now 1:9.10.3.dfsg.P4-8ubuntu1 amd64, Microsoft .NET Framework 4.5.1, 7-Zip 9.20 (x64 edition), libkrb5-26-heimdal/now 1.6 git20131207+dfsg-1ubuntu1.1 amd64, Python 2.7.11, libatk1.0-0/xenial,now 2.18.0-1 amd64, liblapack3/xenial, now 3.6.0-2ubuntu2 amd64, virtual/ssh-0, emacs/xenial,xenialnow 46.1 all, Adobe Actrobat Reader DC 15.010.20060, Java 8 Update 66, Java 7 Update 10, Putty 0.67.0.0, Google Chrome 49.0.2623.112, Python 2.7.11, Mozilla Maintenance Service, sys-apps/kmod-23, libpopt0/trusty,now 1.16-8ubuntu1 amd64, Oracle VM VirtualBox Guest Additions 5.0.10

2. False negatives

   a) OpenVas (35)

      i. Microsoft System CLR Types for SQL Server 2017 CTP2.1 14.0.600.250, Mozilla Maintenance Service 43, Oracle VM VirtualBox Guest Additions 4.1.18, Mozilla Firefox (3.6.28), libxcb-dri3-0/trusty,bow 1.10-2ubuntu1 amd64, virtual/perl-Test-Simple-1.1.2, sys-apps/findutils-4.4.2-r1, Microsoft Visual C++ 2010 x86 Redistributable -10.0.40219, uuid-runtime/now 2.20.1-5.1ubuntu20.7 amd 64, Microsoft SQL Server 2012 (64-bit), Zabbix Agent 2.0.0, dev-perl/HTML-Tagset-3.200.0-r1, dpkg-dev/now 1.18.4ubuntu1.1 all, sys-apps/debianutils-4.7, libc-dev-bin/now 2.19-0ubuntu6.6 amd64, time/trusty, now 1.7-24 amd64, dev-utils/kbuild-0.1.9998_pre20131130-r1, FileZilla Client 3.5.3, Microsoft .NET Framework 4.5.50938, libudev1/now 204-5ubuntu20.15 amd64, Mozilla Maintenance Service, mail-mta/nullmailer-1.13-r5, Microsoft .NET Framework 4.5.1, libkrb5-26-heimdal/now 1.6 git20131207+dfsg-1ubuntu1.1 amd64, Python 2.7.11, libatk1.0-0/xenial,now 2.18.0-1 amd64, liblapack3/xenial, now 3.6.0-2ubuntu2 amd64, Windows Support Tools 5.1.2600.2180, emacs/xenial,xenialnow 46.1 all, Microsoft SQL Server 2012 Policies 11.0.2100.60, Python 2.7.11, Mozilla Maintenance Service, sys-apps/kmod-23, libpopt0/trusty,now 1.16-8ubuntu1 amd64, Oracle VM VirtualBox Guest Additions 5.0.10

   b) Nexpose (6)

      i. Microsoft System CLR Types for SQL Server 2017 CTP2.1, Microsoft Visual C++ 2010 x86 Redistributable -10.0.40219, Zabbix Agent 2.0.0, Windows Support Tools, Microsoft SQL Server 2012 Policies, 7-Zip 9.20

Verified applications, OpenVas

1. True positives (46)

   a) cpe:/a:openbsd:openssh:6.6.1p1, cpe:/a:ntp:ntp:4.2.6p5, cpe:/a:beasts:vsftpd:3.0.2, cpe:/a:microsoft:office_publisher:2013, cpe:/a:perl:perl:5.24.1, cpe:/o:microsoft: windows_server_2008:r2:sp1:x64, cpe:/a:microsoft:ie:6.0.2800.1106, cpe:/a:microsoft: ie:8.0.6001.18702, cpe:/a:microsoft:access:2013, cpe:/a:gnu:gzip:1.6, cpe:/a:microsoft:

onenote:15.0.4569.1503, cpe:/a:don_ho:notepad++:7.5.1, cpe:/a:mozilla:firefox:46.0.1, cpe:/a:gnu:gzip:1.6, cpe:/a:7-zip:7-zip:x64:9.20, cpe:/a:microsoft:rdp:6.1.7601.17514, cpe:/a:oracle:jre:1.7.0:update_10, cpe:/a:libpng:libpng:1.2.54, cpe:/a:isc:bind:9.4.2, cpe:/a:microsoft:access:2013, cpe:/a:microsoft:msn_messenger:4.7.0.41, cpe:/a: microsoft:rdp:5.1.2600.1106, cpe:/a:oracle:jre:1.8.0:update_66, cpe:/a:mozilla:firefox:46.0.1, cpe:/a:gnu:gcc:4.8, cpe:/a:putty:putty:0.64, cpe:/a:gnu:binutils:2.24, cpe:/a: andy: cgi.pm:3.49, cpe:/o:microsoft:windows_server_2008:r2:sp1:x64, cpe:/a:microsoft:rdp: 5.2.3790.3959, cpe:/a:mozilla:firefox:45.0.1, cpe:/a:libpng:libpng:1.2.54, cpe:/a:7-zip:7-zip:9.20, cpe:/a:perl:perl:5.18.2, cpe:/a:oracle:openjdk:1.8.0:update_91, cpe:/a:oracle:mysql, cpe:/a:oracle:jre:1.7.0:update_10, cpe:/a:wireshark:wireshark:2.0.5, cpe:/a:microsoft:ie:6.0.3790.3959, cpe:/a:microsoft:ie:8.0.7601.17514, cpe:/a:microsoft: ie:11.0.9600.16428, cpe:/a:mozilla:firefox:x64:43.0, cpe:/a:7-zip:7-zip:9.20, cpe:/a:microsoft:onenote:15.0.4569.1503, cpe:/a:microsoft:office_publisher:2013, cpe:/a:apache:mod_perl:2.0.4, cpe:/a:adobe:acrobat_reader:15.010.20060

2. False positives (4)

    a) cpe:/a:microsoft:sql_server:2005, cpe:/a:andy_armstrong:cgi.pm:3.49, cpe:/a:rafael_garcia-suarez:safe:2.35, cpe:/a:gnu:gzip:1.2.4

Verified vulnerabilities, OpenVas

1. True positives (37)

    a) CVE-2017-7772, CVE-2017-7786, CVE-2016-3286, CVE-2017-13144, CVE-2018-0810, CVE-2010-1735, CVE-2018-12832, CVE-2016-5191, CVE-2018-18500, CVE-2015-0286, CVE-2015-7545, CVE-2014-1803, CVE-2017-11254, CVE-2018-6096, CVE-2015-0016, CVE-2014-1553, CVE-2018-6031, CVE-2018-0978, CVE-2017-8547, CVE-2017-9346, CVE-2017-5043, CVE-2016-3298, CVE-2012-0449, CVE-2018-5007, CVE-2017-11886, CVE-2018-8470, CVE-2017-7701, CVE-2017-3113, CVE-2017-13015, CVE-2017-5434, CVE-2016-0963, CVE-2018-16009, CVE-2015-6146, CVE-2017-0288, CVE-2016-1049, CVE-2017-5461, CVE-2017-15429

2. False positives (13)

    a) CVE-2016-7632, CVE-2013-3136, CVE-2015-6140, CVE-2017-12177, CVE-2015-6175, CVE-2013-5051, CVE-2018-2657, CVE-2015-2501, CVE-2013-1255, CVE-2016-4476, CVE-2015-2613, CVE-2013-3206, CVE-2015-4173

Verified applications, Nexpose

1. True positives (49)

    a) Mozilla | Firefox|45.0.1, Ubuntu | libxcb-xfixes0|1.11.1-1ubuntu1, Microsoft | .NET Framework 3.5|SP1, Mozilla | Mozilla Maintenance Service|27.0.1, Gentoo | sys-apps/sed|4.2.2, Microsoft | MSXML|6.30.7601.17514, Gentoo | app-portage/eix|0.32.4, Mozilla | Firefox|27.0.1, Ubuntu | systemd|229-4ubuntu6, Ubuntu | python-requests|2.2.1-1ubuntu0.3, Microsoft | .NET Framework 3.5|SP1, Ubuntu | libwmf0.2-7|0.2.8.4-10.5ubuntu1, Ubuntu | biosdevname|0.4.1-0ubuntu6.2, Microsoft | MSXML|8.110.9600.17931, Python Software Foundation | Python 2.7.11|2.7.11150, Ubuntu | speech-dispatcher|0.8.3-1ubuntu3, Microsoft | Access 2013|15.0.4569.1503, Ubuntu | ncurses-bin|5.9+20140118-1ubuntu1, Ubuntu | libasound2-data|1.1.0-0ubuntu1, Adobe | Acrobat Reader DC|15.010.20060, Microsoft | MSXML|4.30.2107.0, Microsoft | .NET Framework 1.1|SP1, Microsoft | OneNote 2013|15.0.4569.1503,

Gentoo | perl-core/Digest-SHA|5.820.0, Simon Tatham | PuTTY|0.67.0.0, Microsoft | Internet Explorer|11.0.9600.17843, Ubuntu | mate-sensors-applet|1.12.1-1, Ubuntu | procps|2:3.3.10-4ubuntu2, Microsoft | Access 2013|15.0.4569.1503, Adobe | Acrobat Reader DC|15.010.20060, No vendor available | Google Update Plugin|1.3.29.5, Microsoft | .NET Framework 2.0|SP2, Microsoft | .NET Framework 3.0|SP2, Microsoft | MDAC|2.8, Microsoft | Lync 2013|15.0.4569.1506, Microsoft | MSXML|6.30.7601.17514, Ubuntu | libmate-slab0|1.12.1-2, Microsoft | MSXML|8.0.7002.0, Ubuntu | libwnck-3-common|3.14.1-2, Microsoft | SQL Server 2012|No version available, Microsoft | Access 2013|15.0.4569.1503, Microsoft | Microsoft Url Search Hook|11.0.9600.17840, Ubuntu | libkrb5-26-heimdal|1.7 git20150920+dfsg-4ubuntu1, VMware, Inc. | VMware Tools|9.4.15.2827462, Google | Chrome|49.0.2623.112, Microsoft | MSXML|6.30.7601.17514, Igor Pavlov | 7-Zip|9.20.00.0, Mozilla | Firefox|27.0.1, Microsoft | .NET Framework 4.5.1 Client Profile|No version available

2. False positives (1)

    a) Microsoft | PowerPoint 2013|15.0.4454.1000

Verified vulnerabilities, Nexpose

1. True positives (43)

    a) CVE-2016-4285, CVE-2015-8779, CVE-2018-1040, CVE-2017-15424, CVE-2019-0755, CVE-2013-1303, CVE-2019-0628, CVE-2012-1527, CVE-2016-4233, CVE-2017-2629, CVE-2016-5297, CVE-2016-7926, CVE-2011-1227, CVE-2014-4081, CVE-2018-0950, CVE-2018-12384, CVE-2018-5143, CVE-2017-13032, CVE-2016-5296, CVE-2015-6085, CVE-2016-7929, CVE-2014-6352, CVE-2015-0355, CVE-2015-0802, CVE-2017-13020, CVE-2016-4191, CVE-2013-3172, CVE-2017-7827, CVE-2017-5115, CVE-2015-1705, CVE-2017-5445, CVE-2018-0846, CVE-2016-7012, CVE-2017-5433, CVE-2017-0248, CVE-2017-0275, CVE-2016-0070, CVE-2015-2444, CVE-2015-5277, CVE-2016-0068, CVE-2018-5040, CVE-2016-5283, CVE-2017-5341

2. False positives (7)

    a) CVE-2017-7601, CVE-2015-0051, CVE-2015-7705, CVE-2018-7169, CVE-2017-2457, CVE-2015-7852, CVE-2014-6504

Collected vulnerabilities, union (OpenVas and Nexpose)

1. True positives (29)

    a) CVE-2017-7768, CVE-2017-7761, CVE-2014-1806, CVE-2018-0796, CVE-2016-2563, CVE-2017-7767, CVE-2009-0511, CVE-2018-10115, CVE-2016-1108, CVE-2016-1077, CVE-2009-2982, CVE-2018-11529, CVE-2016-2826, CVE-2016-4116, CVE-2016-4109, CVE-2016-2335, CVE-2016-4108, CVE-2016-1077, CVE-2016-4093, CVE-2016-4208, CVE-2017-7760, CVE-2018-5996, CVE-2015-1670, CVE-2018-1061, CVE-2014-1806, CVE-2012-0008, CVE-2017-7766, CVE-2016-1102, CVE-2015-2526

2. False negatives (21)

    a) CVE-2018-20681, CVE-2006-4138, CVE-2014-2441, CVE-2015-4813, CVE-2006-0745, CVE-2006-3014, CVE-2018-1000035, CVE-2017-1000158, CVE-2010-3493, CVE-2014-7185, CVE-2006-3014, CVE-2016-4338, CVE-2018-1060, CVE-2010-3190, CVE-2018-18064, CVE-2014-0253, CVE-2018-19666, CVE-2015-6607, CVE-2008-3012, CVE-2010-1634, CVE-2016-1104

Verified vulnerabilities, union (OpenVas and Nexpose)

1. True positives (80)

    a) CVE-2017-7772, CVE-2017-7786, CVE-2016-3286, CVE-2017-13144, CVE-2018-0810, CVE-2010-1735, CVE-2018-12832, CVE-2016-5191, CVE-2018-18500, CVE-2015-0286, CVE-2015-7545, CVE-2014-1803, CVE-2017-11254, CVE-2018-6096, CVE-2015-0016, CVE-2014-1553, CVE-2018-6031, CVE-2018-0978, CVE-2017-8547, CVE-2017-9346, CVE-2017-5043, CVE-2016-3298, CVE-2012-0449, CVE-2018-5007, CVE-2017-11886, CVE-2018-8470, CVE-2017-7701, CVE-2017-3113, CVE-2017-13015, CVE-2017-5434, CVE-2016-0963, CVE-2018-16009, CVE-2015-6146, CVE-2017-0288, CVE-2016-1049, CVE-2017-5461, CVE-2017-15429, CVE-2016-4285, CVE-2015-8779, CVE-2018-1040, CVE-2017-15424, CVE-2019-0755, CVE-2013-1303, CVE-2019-0628, CVE-2012-1527, CVE-2016-4233, CVE-2017-2629, CVE-2016-5297, CVE-2016-7926, CVE-2011-1227, CVE-2014-4081, CVE-2018-0950, CVE-2018-12384, CVE-2018-5143, CVE-2017-13032, CVE-2016-5296, CVE-2015-6085, CVE-2016-7929, CVE-2014-6352, CVE-2015-0355, CVE-2015-0802, CVE-2017-13020, CVE-2016-4191, CVE-2013-3172, CVE-2017-7827, CVE-2017-5115, CVE-2015-1705, CVE-2017-5445, CVE-2018-0846, CVE-2016-7012, CVE-2017-5433, CVE-2017-0248, CVE-2017-0275, CVE-2016-0070, CVE-2015-2444, CVE-2015-5277, CVE-2016-0068, CVE-2018-5040, CVE-2016-5283, CVE-2017-5341

2. False positives (20)

    a) CVE-2016-7632, CVE-2013-3136, CVE-2015-6140, CVE-2017-12177, CVE-2015-6175, CVE-2013-5051, CVE-2018-2657, CVE-2015-2501, CVE-2013-1255, CVE-2016-4476, CVE-2015-2613, CVE-2013-3206, CVE-2015-4173, CVE-2017-7601, CVE-2015-0051, CVE-2015-7705, CVE-2018-7169, CVE-2017-2457, CVE-2015-7852, CVE-2014-6504

Collected vulnerabilities, intersection (OpenVas and Nexpose)

1. True positives (15)

    a) CVE-2017-7768, CVE-2017-7761, CVE-2014-1806, CVE-2018-0796, CVE-2017-7767, CVE-2009-0511, CVE-2018-10115, CVE-2009-2982, CVE-2016-2826, CVE-2017-7760, CVE-2015-1670, CVE-2018-1061, CVE-2014-1806, CVE-2017-7766, CVE-2015-2526

2. False negatives (35)

    a) CVE-2018-20681, CVE-2006-4138, CVE-2014-2441, CVE-2016-2563, CVE-2015-4813, CVE-2006-0745, CVE-2006-3014, CVE-2018-1000035, CVE-2017-1000158, CVE-2016-1108, CVE-2010-3493, CVE-2016-1077, CVE-2014-7185, CVE-2006-3014, CVE-2016-4338, CVE-2018-11529, CVE-2018-1060, CVE-2016-4116, CVE-2016-4109, CVE-2016-2335, CVE-2016-4108, CVE-2016-1077, CVE-2010-3190, CVE-2016-4093, CVE-2016-4208, CVE-2018-18064, CVE-2018-5996, CVE-2014-0253, CVE-2018-19666, CVE-2015-6607, CVE-2012-0008, CVE-2016-1102, CVE-2008-3012, CVE-2010-1634, CVE-2016-1104

Verified vulnerabilities, intersection (OpenVas and Nexpose)

1. True positives (61)

    a) CVE-2017-7772, CVE-2017-7786, CVE-2016-3286, CVE-2017-13144, CVE-2018-0810, CVE-2018-12832, CVE-2016-5191, CVE-2018-18500, CVE-2015-7545, CVE-2014-1803, CVE-2017-11254, CVE-2018-6096, CVE-2015-0016, CVE-2014-1553, CVE-2018-6031, CVE-2018-0978, CVE-2017-9346, CVE-2017-5043, CVE-2016-3298, CVE-2012-0449, CVE-2018-8470, CVE-2017-7701, CVE-2017-3113, CVE-2017-13015, CVE-2017-5434, CVE-2016-0963, CVE-2018-16009, CVE-2015-6146, CVE-2016-1049, CVE-2017-5461,

CVE-2017-15429, CVE-2015-8779, CVE-2018-1040, CVE-2017-15424, CVE-2019-0628, CVE-2012-1527, CVE-2016-4233, CVE-2016-5297, CVE-2011-1227, CVE-2014-4081, CVE-2018-0950, CVE-2018-5143, CVE-2016-5296, CVE-2015-6085, CVE-2014-6352, CVE-2015-0355, CVE-2016-4191, CVE-2013-3172, CVE-2017-7827, CVE-2017-5115, CVE-2015-1705, CVE-2017-5445, CVE-2018-0846, CVE-2017-5433, CVE-2017-0275, CVE-2016-0070, CVE-2015-2444, CVE-2016-0068, CVE-2018-5040, CVE-2016-5283, CVE-2017-5341

2. False positives (13)

   a) CVE-2013-3206, CVE-2015-2613, CVE-2016-4476, CVE-2015-2501, CVE-2018-2657, CVE-2013-5051, CVE-2015-6175, CVE-2017-12177, CVE-2015-6140, CVE-2016-7632, CVE-2014-6504, CVE-2017-2457, CVE-2015-7705

9.3 Inconsistencies evaluation

See table 9.2, 9.3, 9.4 and 9.5.

Network	Computer	Apps in exp 1 not in exp 2	Apps in exp 2 not in exp 1
Process	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:24805	cpe:/a:mozilla:thunderbird:12997, cpe:/a:jquery:jquery:3.2.1
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:8807	cpe:/a:mozilla:thunderbird:32394, cpe:/a:jquery:jquery:3.2.1
	Ubuntu 16.04	cpe:/a:io-socket-ssl:io-socket-ssl:2.024, cpe:/a:mozilla:thunderbird:24249, cpe:/a:rafael_garcia-suarez:safe:2.39, cpe:/a:perl:perl:5.22.1, cpe:/a:perl:archive_tar:2.04	cpe:/a:mozilla:thunderbird:16024, cpe:/a:jquery:jquery:3.2.1
	Ubuntu 16.04	cpe:/a:io-socket-ssl:io-socket-ssl:2.024, cpe:/a:mozilla:thunderbird:4507, cpe:/a:rafael_garcia-suarez:safe:2.39, cpe:/a:perl:perl:5.22.1, cpe:/a:perl:archive_tar:2.04	cpe:/a:mozilla:thunderbird:28068, cpe:/a:jquery:jquery:3.2.1
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:17100	cpe:/a:avahi:avahi:0.6.32, cpe:/a:mozilla:thunderbird:8575, cpe:/a:jquery:jquery:3.2.1

Table 9.2: Inconsistencies scans 1-2, OpenVas

Network	Computer	Apps in scan 1 not in scan 2	Apps in scan 2 not in scan 1
Process	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:24805	cpe:/a:jquery:jquery:3.2.1, cpe:/a:mozilla:thunderbird:20108
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:8807	cpe:/a:jquery:jquery:3.2.1, cpe:/a:mozilla:thunderbird:15611
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:24249, cpe:/a:avahi:avahi:0.6.32	cpe:/a:jquery:jquery:3.2.1, cpe:/a:mozilla:thunderbird:32210
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:4507	cpe:/a:jquery:jquery:3.2.1, cpe:/a:mozilla:thunderbird:12364
	Ubuntu 16.04	cpe:/a:mozilla:thunderbird:17100	cpe:/a:jquery:jquery:3.2.1, cpe:/a:avahi:avahi:0.6.32, cpe:/a:mozilla:thunderbird:24157

Table 9.3: Inconsistencies scans 1-3, OpenVas

Network	Computer	Vuls in scan 1 not in scan 2	Vuls in scan 2 not in scan 1
Internal	All computers	-	CVE-1999-0524
DMZ	All computers	-	CVE-1999-0524
Process	All computers	-	CVE-1999-0524

Table 9.4: Inconsistencies scans 1-2, Nexpose

Network	Computer	Vuls in scan 1 not in scan 2	Vuls in scan 2 not in scan 1
Internal	Windows 7	-	CVE-1999-0524
DMZ	All computers	-	CVE-1999-0524
Process	All computers	-	-

Table 9.5: Inconsistencies scans 1-3, Nexpose

9.4 Experiment

Control Panel uninstall

1. Uninstalled applications (13)
    a) 7-Zip 15.14, Adobe Acrobat Reader DC, Adobe Flash Player 21 ActiveX, Adobe Flash Player 21 NPAPI, FileZilla Client 3.16.1, Google Chrome, Java 7 Update 10, Microsoft Office Professional Plus 2013, Mozilla Firefox 45.0.1, Mozilla Maintenance Service, PuTTy, Python 2.7 pywin32-220, VLC media player
2. Left installed? (2)
    a) Oracle VM VirtualBox Guest Additions, Python 2.7.11

"/usr/bin" folders emptied

Path: /usr/bin

1. All subfolders removed (1980)

"/usr/sbin" folders cleared

Path: /usr/sbin

1. All subfolders removed (211)

"Uninstall" registry key cleared

Registry key: HKEY_LOCAL_MACHINESoftware MicrosoftWindowsCurrentVersionUninstall

1. Deleted (41)
    a) {90150000-0015-0409-0000-0000000FF1CE}, {90150000-0016 -0409-0000-0000000FF1CE}, {90150000-0018-0409-0000-0000000FF1CE}, {90150000-0019-0409-0000-0000000FF1CE}, {90150000-001A-0409-0000-0000000FF1CE}, {90150000-001B-0409-0000-0000000FF1CE}, {90150000-001F-0409-0000-0000000FF1CE}, {90150000-001F-040C-0000-0000000FF1CE}, {90150000-002C-0409-0000-0000000FF1CE}, {90150000-0044-0409-0000-0000000FF1CE}, {90150000-006E-0409-0000-0000000FF1CE}, {90150000-0090-0409-0000-0000000FF1CE}, {90150000-00A1-0409-0000-0000000FF1CE}, {90150000-00BA-0409-0000-0000000FF1CE}, {90150000-00E1-0409-0000-0000000FF1CE}, {90150000-00E2-0409-0000-0000000FF1CE}, {90150000-0117-0409-0000-0000000FF1CE}, {90150000-012B-0409-0000-0000000FF1CE}, 7-Zip,

AddressBook, Adobe Flash Player ActiveX, Adobe Flash Player NPAPI, Connection Manager, DirectDrawEx, DXM_Runtime, FileZilla Client, Fontcore, Google Chrome, IE40, IE4Data, IE5BAKEX, IEData, MobileOptionPack, Mozilla Firefox 45.0.1 (x86 en-US), MozillaMaintenanceService, MPlayer2, Office15.PROPLUS, Oracle VM VirtualBox Guest Additions, SchedulingAgent, WIC, VLC media player

2. Left (0)

Program folders moved

Path program files: Computer->Local Disc(C)->Program Files Path new folder: Desktop->tmp

1. Moved (24)

    a) 7-Zip, Adobe, DVD Maker, FileZilla FTP Client, Internet Explorer, Java, Microsoft Analysis Services, Microsoft Office, Microsoft SQL Server, Microsoft.NET, Mozilla Firefox, Mozilla Maintenance Service, MSBuild, Oracle, PuTTy, Reference Assemblies, VideoLan, Windows Defender, Windows Journal, Windows Mail, Windows Media Player, Windows NT, Windows Photo Viewer, Windows Portable Devices

2. Left (3)

    a) Common Files, Google, Windows Sidebar

Services disabled

1. Disabled (135)

    a) ActiveX Installer (AxInstSV), Adaptive Brightness, Adobe Acrobat Update Service, Adobe Flash Player Update Service, Application Experience, Application Identity, Application Information, Application Layer Gateway Service, Application Management, Background Intelligent Transfer Service, Base Filtering Engine, BitLocker Drive Encryption Service, Block Level Backup Engine Service, Bluetooth Support Service, BranchCache, Certificate Propagation, CNG Key Isolation, COM+ Event System, COM+ System Application, Computer Browser, Credential Manager, Cryptographic Services, Desktop Window Manager Session Manager, Diagnostic Policy Service, Diagnostic Service Host, Diagnostic System Host, Disk Defragmenter, Distributed Link Tracking Client, Distributed Transaction Coordinator, DNS Client, Encrypting File System (EFS), Extensible Authentication Protocol, Fax, Function Discovery Provider Host, Function Discovery Resource Publication, Health Key and Certificate Management, HomeGroup Listener, HomeGroup Provider, Human Interface Device Access, IKE and AuthIP IPsec Keying Modules, Interactive Services Detection, Internet Connection Sharing (ICS), Internet Explorer ETW Collector Service, IP Helper, IPsec Policy Agent, KtmRm for Distributed Transaction Coordinator, Link-Layer Topology Discovery Mapper, Media Center Extender Service, Microsoft .NET Framework NGEN v2.0.50727_X86, Microsoft iSCSI Initiator Service, Microsoft Software Shadow Copy Provider, Mozilla Maintenance Service, Multimedia Class Scheduler, Net.Tcp Port Sharing Service, Network Access Protection Agent, Network Connections, Network List Service, Network Location Awareness, Office Source Engine, Office Software Protection Platform, Offline Files, Parental Controls, Peer Name Resolution Protocol, Peer Networking Grouping, Peer Networking Identity Manager, Performance Logs Alerts, PnP-X IP Bus Enumerator, PNRP Machine Name Publication Service, Portable Device Enumerator Service, Print Spooler, Problem Reports and Solutions Control Panel Support, Program Compatibility Assistant Service, Protected Storage, Quality Windows Audio Video Experience, Remote Access Auto Connection Manager, Remote Access Connection Manager, Remote Desktop Configuration,

Remote Desktop Services, Remote Desktop Services UserMode Port Redirector, Remote Procedure Call (RPC) Locator, Routing and Remote Access, Secondary Logon, Secure Socket Tunneling Protocol Service, Security Center, Shell Hardware Detection, Smart Card, Smart Card Removal Policy, SNMP Trap, Software Protection, SPP Notification Service, SSDP Discovery, Storage Service, Superfetch, System Event Notification Service, TCP/IP NetBIOS Helper, Telephony, Themes, Thread Ordering Server, Tjänsten Google Update (gupdate), Tjänsten Google Update (gupdatem), TPM Base Services, UPnP Device Host, User Profile Service, WebClient, Windows Audio, Windows Audio Endpoint Builder, Windows Backup, Windows Biometric Service, Windows CardSpace, Windows Color System, Windows Connect Now - Config Registrar, Windows Defender, Windows Driver Foundation - User-mode Driver Framework, Windows Error Reporting Service, Windows Event Collector, Windows Firewall, Windows Font Cache Service, Windows Image Acquisition (WIA), Windows Installer, Windows Management Instrumentation, Windows Media Center Receiver Service, Windows Media Center Scheduler Service, Windows Media Player Network Sharing Service, Windows Modules Installer, Windows Presentation Foundation Font Cache 3.0.0.0, Windows Remote Management (WS-Management), Windows Search, Windows Time, Windows Update, WinHTTP Web Proxy Auto-Discovery Service, Wired AutoConfig, Virtual Disk, WLAN AutoConfig, WMI Performance Adapter, Volume Shadow Copy

2. Not disabled (19)

    a) DCOM Server Process Launcher, DHCP Client, Group Policy Client, Netlogon, Network Store Interface Service, Plug and Play, Power, Remote Procedure Call (RPC), Remote Registry, RPC Endpoint Mapper, Security Accounts Manager, Server, Tablet PC Input Service, Task Scheduler, Windows Event Log, VirtualBox Guest Additions Service, Workstation, WWAN AutoConfig

Value names in "Known DLLs" registry key deleted

Key: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDLLs

1. Deleted (29)

    a) advapi32, clbcatq, COMDLG32, DifxApi, DllDirectory, gdi32, IERTUTIL, IMAGEHLP, IMM32, kernel32, LPK, MSCTF, MSVCRT, NORMALIZ, NSI, ole32, OLEAUT32, PSAPI, rpcrt4, sechost, Setupapi, SHELL32, SHLWAPI, URLMON, user32, USP10, WININET, WLDAP32, WS2_32

2. Left (1)

    a) (Default)

Subkeys "Service" registry key deleted

Registry key:

HKLM\SYSTEM\CurrentControlSet\Services

Subkeys "App paths" registry key deleted

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths

1. Deleted (37)

a) 7zFM.exe, AcroRd32.exe, chrome.exe, cmmgr32.exe, excel.exe, filezilla.exe, firefox.exe, GROOVE.EXE, IEDIAG.EXE, IEDIAGCMD.EXE, IEXPLORE.EXE, infopath.exe, install.exe, javaws.exe, Lync.exe, migwiz.exe mplayer2.exe, MSACCESS.EXE, MsoHtmEd.exe, msoxmled.exe, MSPUB.EXE, OneNote.exe, OUTLOOK.EXE, pbrush.exe, powerpnt.exe, PowerShell.exe, setup.exe, sidebar.exe, table30.exe, wab.exe, wabmig.exe, Winword.exe, wmplayer.exe, WORDPAD.EXE, WRITE.EXE, vstoee.dll

2. Left (5)

a) dvdmaker.exe, Journal.exe, mip.exe, SnippingTool.exe, TapTip.exe

Bibliography

[1] Anurag Akkiraju, David Gabay, Halim Burak Yesilyurt, Hidayet Aksu, and Selcuk Uluagac. "Cybergrenade: Automated Exploitation of Local Network Machines via Single Board Computers". In: *14th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. Miami, 2017, pp. 580–584. ISBN: 9781538623237. DOI: 10.1109/MASS.2017.95.

[2] *AN OVERVIEW OF VULNERABILITY SCANNERS*. Tech. rep. The Government of the Hong Kong Special Administrative Region, 2008.

[3] Timothy M Braje. "Advanced Tools for Cyber Ranges". In: *LINCOLN LABORATORY JOURNAL* 22.1 (2016), pp. 24–32.

[4] Michael Buckland and Fredric Gey. "The relationship between Recall and Precision". In: *Journal of the American Society for Information Science* 45.1 (1994), pp. 12–19. DOI: 10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[5] Tom Carpenter. *Microsoft Windows operating system essentials*. John Wiley & Sons, 2012. ISBN: 9781118227688.

[6] *Classification: Accuracy*. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy (visited on 06/10/2019).

[7] *CRATE - Cyber Range And Training Environment*. 2019. URL: https://www.foi.se/en/foi/resources/crate---cyber-range-and-training-environment.html (visited on 06/10/2019).

[8] Suresh Damodaran and Kathy Smith. *CRIS Cyber Range Lexicon Version 1.0*. Tech. rep. MIT Lincoln Laboratory. URL: https://www.researchgate.net/publication/316322192.

[9] Nor Izyani Daud, Khairul Azmi Abu Bakar, and Mohd Shafeq Md Hasan. "A case study on web application vulnerability scanning tools". In: *Science and Information Conference, SAI*. London, 2014, pp. 595–600. ISBN: 9780989319317. DOI: 10.1109/SAI.2014.6918247.

[10] Jon Davis and Shane Magrath. *A Survey of Cyber Ranges and Testbeds*. Tech. rep. DEFENCE SCIENCE, TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) CYBER, and ELECTRONIC WARFARE DIV.

[11] Frank Doelitzscher, Christian Fischer, Denis Moskal, Christoph Reich, Martin Knahl, and Nathan Clarke. "Validating Cloud Infrastructure Changes By Cloud Audits". In: *IEEE Eighth World Congress on Services*. Honolulu, 2012, pp. 377–384. DOI: 10.1109/SERVICES.2012.12.

[12] Jodé Fonseca, Vieira Marco, and Madeira Henrique. “Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks”. In: *13th IEEE International Symposium on Pacific Rim Dependable Computing*. Melbourne, 2007, pp. 365–372. DOI: 10.1109/PRDC.2007.55.

[13] *Free Nexpose Community 1-Year Trial*. URL: https://www.rapid7.com/info/nexpose-community/ (visited on 06/10/2019).

[14] Ionel Gordin, Adrian Graur, Alin Potorac, and Doru Balan. “Security assessment of OpenStack cloud using outside and inside software tools”. In: Suceava, Romania: 2018 International Conference on Development and Application Systems, 2018, pp. 170–174. ISBN: 9781538614952.

[15] *Greenbone Vulnerability Manager*. URL: https://github.com/greenbone/gvmd (visited on 06/10/2019).

[16] Hannes Holm. “Performance of automated network vulnerability scanning at remediating security issues”. In: *Computers and Security* 31.2 (Mar. 2012), pp. 164–175. ISSN: 01674048. DOI: 10.1016/j.cose.2011.12.014.

[17] Hannes Holm, Teodor Sommestad, Jonas Almroth, and Mats Persson. “A quantitative evaluation of vulnerability scanning”. In: *Information Management and Computer Security* 19.4 (2011), pp. 231–247. ISSN: 09685227. DOI: 10.1108/09685221111173058.

[18] *Locked Shields*. URL: https://ccdcoe.org/exercises/locked-shields/ (visited on 06/18/2019).

[19] Raymond Lukanta, Yudistira Asnar, and A Imam Kistijantoro. “A Vulnerability Scanning Tool for Session Management Vulnerabilities”. In: Bandung: International Conference on Data and Software Engineering, 2014. DOI: 10.1109/ICODSE.2014.7062682.

[20] Osamah M Al-Matari, Iman M A Helal, Sherif A Mazen, and Sherif Elhennawy. “Cybersecurity Tools for IS Auditing”. In: *The 6th International Conference on Enterprise Systems*. Limassol, Cyprus, 2018. DOI: 10.1109/ES.2018.00040.

[21] Sarang Na, Taeeun Kim, and Hwankuk Kim. “Service identification of Internet-connected devices based on common platform enumeration”. In: *Journal of Information Processing Systems* 14.3 (2018), pp. 740–754. DOI: 10.3745/JIPS.03.0098.

[22] *National Vulnerable Database*. 2019. URL: https://nvd.nist.gov/ (visited on 06/10/2019).

[23] Christopher Negus. *Linux Bible*. John Wiley & Sons, Incorporated, 2015. ISBN: 9781118999882.

[24] Wu Qianqian and Liu Xiangjun. “Research and design on Web application vulnerability scanning service”. In: *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*. Beijing, China, 2014, pp. 671–674. ISBN: 9781479932788. DOI: 10.1109/ICSESS.2014.6933657.

[25] Enrico Russo, Gabriele Costa, and Alessandro Armando. “Scenario design and validation for next generation cyber ranges”. In: *2018 IEEE 17th International Symposium on Network Computing and Applications*. Cambridge, MA, USA: Institute of Electrical and Electronics Engineers Inc., Nov. 2018. ISBN: 9781538676592. DOI: 10.1109/NCA.2018.8548324.

[26] Yaroslav Stefinko, Andrian Piskozub, and Roman Banakh. “Manual and Automated Penetration Testing, Benefits and Drawbacks.” In: *13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*. Lviv, Ukraine, 2016, pp. 488–491. DOI: 10.1109/TCSET.2016.7452095.

[27] *Swedish Defence Research Agency*. URL: https://www.foi.se/en/foi.html (visited on 06/10/2019).

[28] *The Python Standard Library*. URL: https://docs.python.org/2/library/random.html (visited on 06/10/2019).

[29] Karthik Venkateswaran. *Consistency vs Accuracy*. 2015. URL: https://www.linkedin.com/pulse/consistency-vs-accuracy-karthik-venkateswaran/ (visited on 06/10/2019).