OULUN YLIOPISTO
UNIVERSITY of OULU

DEGREE PROGRAMME IN INFORMATION NETWORKS

Pasi Keski-Korsu

Automated port scanning and security testing on a single network host

Master's Thesis
Degree Programme in Information Networks
April 2016

Keski-Korsu P. (2016) Automated port scanning and security testing on a single network host. University of Oulu, Degree Programme in Information Networks. Master's Thesis, 69 p. 3 Appendices.

ABSTRACT

Black-box security testing is divided into five phases: reconnaissance, scanning, exploitation, post exploitation and reporting. There are many tools and methods to perform security testing in the exploitation and post-exploitation phases. Therefore, the first two steps are crucial to execute properly to narrow down different options to do security testing. In the scanning phase, the penetration tester's goal is to gather as much information as possible from the system under test. One task is to discover open network ports and used network protocols. Nmap is a port scanning tool to check network port states and what network protocols target host supports. Nmap's different port scanning techniques should be used to obtain comprehensive results of port states. The results from different scanning techniques have to be combined so that port state assignments have to be allocated to trusted and untrusted assignments.

After port scanning has been executed, the actual software security testing can begin. This testing can be automated to begin right after port scanning. Automated tests are started of services that run behind open ports that have been discovered in port scanning. The Nmap scripting engine also has a module to execute general scripts to gather more information on the system under test. Another tool, Nikto, is implemented to test services that use Hypertext Transfer Protocol (HTTP).

Port scanning and automated testing is time consuming, when scanning and testing is executed comprehensively. Sometimes it is crucial to obtain results in a short time, so there should be options on broadness of scanning and testing. Comprehensive scanning and testing may produce large amounts of scattered information so reporting of the results should be brief and clear to help penetration tester's work. The performance of the scanning and testing implementation is evaluated by testing a single network host and flexibility is validated by running the scanning and testing on other network hosts.

Keywords: penetration testing, Nmap, SECURED project, NED, network edge device

Keski-Korsu P. (2016) Yksittäisen verkkolaitteen automatisoitu porttiskannaus ja tietoturvatestaus. Oulun yliopisto, informaatioverkostojen koulutusohjelma. Diplomityö, 69 s. 3 liitettä.

TIIVISTELMÄ

Black-box-tietoturvatestaus on jaettu viiteen vaiheeseen: tiedustelu, skannaus, hyödyntäminen, hyödyntämisen jälkeiset toimet ja raportointi. On olemassa paljon työkaluja ja metodeja tietoturvatestauksen tekemiseen hyödyntämisvaiheessa ja hyödyntämisen jälkeisissä toimissa. Tämän vuoksi kaksi ensimmäistä vaihetta on tärkeä suorittaa huolellisesti, jotta eri tietoturvatestausvaihtoehtoja voidaan vähentää. Skannausvaiheessa tietoturvatestaajan päämäärä on kerätä mahdollisimman paljon tietoa testikohteesta. Yksi tehtävä tässä vaiheessa on löytää avoimia verkkoportteja ja käytettyjä IP-protokollia. Nmap on porttiskannaustyökalu, jonka avulla voidaan selvittää verkkoporttien tilat sekä käytetyt verkkoprotokollat. Nmap sisältää erilaisia porttiskannaustekniikoita, joita tulee käyttää kattavien skannaustulosten saamiseksi. Eri skannaustekniikoista pitää yhdistellä tuloksia, joten skannaustekniikoiden antamat luokitukset tulee jakaa luotettaviin ja ei-luotettaviin tuloksiin.

Kun porttiskannaus on suoritettu, varsinainen tietoturvatestaus voi alkaa. Testauksen voi automatisoida alkamaan heti porttiskannauksen jälkeen. Automaatiotestit ajetaan palveluihin, jotka toimivat avoimien porttien takana. Avoimet portit on tutkittu porttiskannausvaiheessa. Nmapin skriptityökalu sisältää myös moduulin, joka suorittaa yleisiä testejä testikohteeseen, millä saadaan lisätietoa testattavasta kohteesta. Toinen testaustyökalu, Nikto, on implementoitu testaamaan palveluja, jotka käyttävät Hypertext Transfer Protokollaa (HTTP).

Porttiskannaus ja automatisoitu tietoturvatestaus vie aikaa, kun skannaus ja testaus suoritetaan kokonaisvaltaisesti. Joskus on kuitenkin tärkeää saada tuloksia lyhyessä ajassa, joten testaajalla tulisi olla eri laajuisia skannaus- ja testausvaihtoehtoja. Kokonaisvaltainen skannaus ja testaus voi tuottaa suuren määrän hajallaan olevaa tietoa, joten tulokset pitää raportoida lyhyesti ja selkeästi, jotta penetraatiotestaajan työ helpottuu. Skannaus- ja testausohjelman toimintakyky arvioidaan skannaamalla yksittäinen verkkolaite ja joustavuus muihin ympäristöihin varmistetaan skannaamalla ja testaamalla useampi verkkolaite.

Avainsanat: penetraatiotestaus, Nmap, SECURED projekti, NED, network edge device

TABLE OF CONTENTS

FOREWORD

This thesis has been written and developed while working at VTT Technical Research Centre of Finland Ltd.

I would like to thank Professor Juha Röning and Thomas Schaberreiter for being the supervisor and the second examiner respectively for this thesis. For technical supervision I would like to thank Jouni Hiltunen from VTT. Jouni's guidance has been very useful in helping me understand security testing both in the big picture and with practical testing details. Also, I would like to thank Jarkko Kuusijärvi, the project manager on the SECURED project at VTT. Feedback from all these people has been pushing this work forward and has helped me at difficult moments.

I would also like to thank my friends and fellow students who have given mental support and encouragement to write this thesis.

Oulu, 25.4.2016

Pasi Keski-Korsu

ABBREVIATIONS

ACK	Acknowledgment field significant flag in TCP packet
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common vulnerabilities and exposures
DNS	Domain Name System
ECT	Explicit Connection Target
FIN	No more data from sender flag in TCP packet
GUI	Graphical user interface
HAD	Hybrid Application Detection
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Messaging Protocol
IDS	Intrusion detection system
IP	Internet Protocol
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
MAC	Media access control
NED	Network Edge Device
NSE	Nmap Scripting Engine
OS	Operating system
OWASP	Open Web Application Security Project
PSA	Personal Security Application
PSC	Personal Security Controller
PSH	Push Function in TCP packet
RFC	Request for comments
RST	Reset the connection flag in TCP packet
RTT	Round trip time
SDLC	Software development life cycle
SECURED	SECURity at the network EDge
SNMP	Simple network management protocol
SSH	Secure shell
SUT	System Under Test
SYN	Synchronize sequence numbers flag in TCP packet
TCP	Transmission control protocol
TTL	Time to live
TVD	Trusted Virtual Domain
UDP	User datagram protocol
URG	Urgent Pointer field significant flag in TCP packet
VPN	Virtual private network
XML	Extensible Markup Language

1. INTRODUCTION

One important issue in network security is software vulnerabilities [1]. Vulnerabilities can be described as “flaws in applications that allow attackers to do something malicious” [2] or “a type of bug that can be used by an attacker to alter the intended operation of the software in a malicious way” [3]. Vulnerabilities can expose web sites, systems and even networks to an attacker to carry out unauthorized activities. An attacker might gain access to sensitive information such as credit card numbers, run their own code in the target system to install malicious software or cause denial of service so as to sabotage the system.

One way to prevent these malicious activities is black-box software security testing, which is also called penetration testing. As in black-box testing, in penetration testing the tester has no internal information about the system under test (SUT). Therefore, the tester acts as a real attacker and uses the same techniques to compromise the system [4]. In this way, it is possible to examine the system’s outer behaviour.

Penetration testing can be divided into different phases: reconnaissance, scanning, exploitation, post exploitation and reporting [5]. When SUT is considered as black box, gathering information from the system is an essential part of the testing process. The tester has to become familiar with the system and learn the behaviour of different interfaces. One part of this information gathering is scanning for open network ports to get actual attack surfaces. Comprehensive port scanning can give the tester a lot of information about SUT, and it is a basis for how the tester starts to create security test plan.

One widely used tool for port scanning is Nmap [6]. There are many different methods to perform port scanning with it. For example, port scanning can be customized with six Transmission Control Protocol (TCP) heading’s control bits, which makes 64 different scanning possibilities only with the TCP header’s control bits. The tester has to choose sufficient methods for scanning. Choosing the methods and running them manually can be time consuming and scattered scanning results can be difficult to analyse, store and report. Additionally Nmap has version detection, operating system detection and its own Nmap Scripting Engine (NSE) to interrogate system further after actual scanning.

The scope of this thesis is to automate the whole port scanning process and create a security test plan automatically based on the scanning of a single network host. The goal is to create logic to combine scanning results from different scanning methods, automate version and Operating System (OS) detection, and start security tests so as to automatically test the system. All results should be reported in a clear manner so that the result sheet can be used as a part of the security testing report. Also, using external databases such as Common vulnerabilities and exposures (CVE) should be considered.

The main goal of port scanning is to obtain comprehensive scanning results from SUT: after running the scan, it is not necessary to run any other port scans. This kind of comprehensive scan that is able to gather a massive amount of information can be time-consuming. Therefore it is crucial to develop the automation in such a manner that it should be helpful to the penetration tester: it should be possible to acquire some results in a short time so that the tester is able to start to execute tests and wait for comprehensive scanning results.

This implementation is going to be tested and evaluated in SECURity at the network EDge project (SECURED) [7]. The SECURED project's main idea is to offload security controls from the user device to external Network Edge Device (NED). The scanning and testing implementation is going to be developed to act as an individual component inside NED in order to scan NED's inner components.

2. SOFTWARE SECURITY TESTING

It is possible to approach software security in many ways: security requirements, risk analysis, penetration testing and maintenance after security break for example [8]. The last-mentioned approach is a situation that should be avoided. It is important to have security thinking at all times in the software development life cycle (SDLC) in order to avoid security breaks after the software has been released.

2.1. OWASP testing guide

Open Web Application Security Project (OWASP) is a security testing framework for web applications [9]. It presents a model that emphasizes performing security testing in all phases of SDLC. Therefore penetration testing is only one technique to perform security testing. Other techniques are manual inspection, threat modelling and code review which are implemented in different phases in SDLC. Phases and their positions in SDLC are shown in Figure 1. Although OWASP is focused on web applications, testing techniques can be applied to security testing in general.

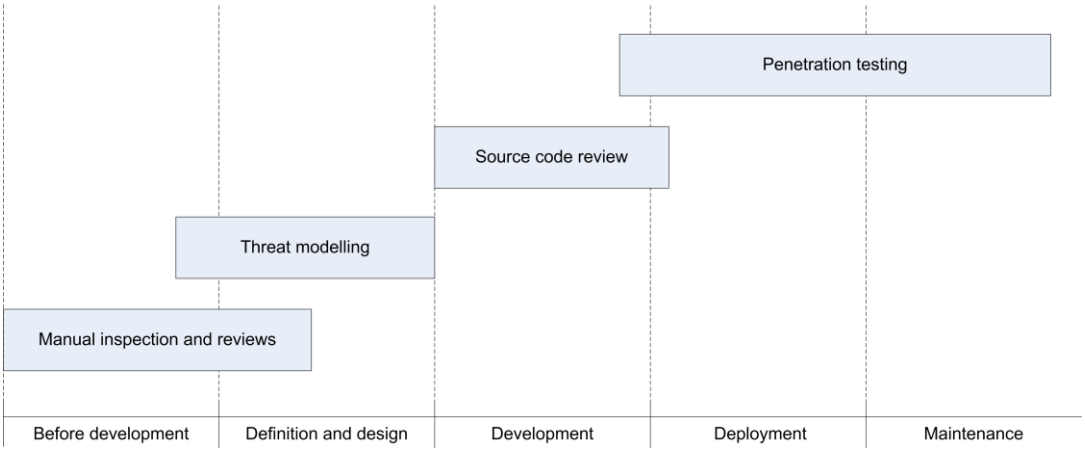

Figure 1. Different testing techniques in SDLC

Manual inspection and reviews is a testing technique implemented in the early phase of SDLC. It relates to making secure technical decisions and determining what security concerns are involved in the project. These inspections give a broad picture of the whole SDLC in the security point of view. Human reviews dig in the developers' mind how different parts of software work and how they are implemented. With these reviews, it is ensured that people involved in the project have security thinking in their head.

Threat modelling is mapping different scenarios of the security threats that developed software might face. A simple model for developing a threat model is to map it with five approaches: decomposing the application, defining and classifying the assets, exploring potential vulnerabilities, exploring potential threats and creating mitigation strategies.

Reviewing source code can be an effective testing method when the tester has access to the developed code. Still, it requires high technical skills on the part of the tester and some security issues are difficult to find just by looking at the code.

Inspecting the source code is white box testing so it is not possible to do this to closed source applications, because there is no access to the source code.

Penetration testing is a black box testing method for software security. Since penetration testing is a black box method, the tester does not have information about the inner workability of the test target. It usually contains an approach where the tester mimics the actions of a real-world security attacker and launches real security attacks [4]. Penetration testing is usually used in the late phase of software development.

2.2. Penetration testing process

There are many ways to determine steps to the penetration testing process [5, 10–12]. Processes vary a little in details, but most of them follow the same five-step pattern [10] shown in Figure 2. The steps are reconnaissance, scanning, exploitation, post exploitation and reporting. It is worth noting that penetration testing does not start by connecting immediately to the target system or network.

First, there has to be information gathering from public resources and discussion with the client. One main issue is to specify the scope of testing. After that the actual testing process can be started by scanning the target system or network so as to gather technical information. The actual exploitation or vulnerability analysis and testing of the system is the third step. If vulnerabilities are found, it is important to determine how the vulnerability can be exploited. In post exploitation, the criticality of the vulnerability can be determined. Finally test results should be reported. [10]

Figure 2. Penetration testing process

2.2.1. Reconnaissance

Reconnaissance, or information gathering, is one of the most overlooked steps in the penetration testing process, although its value for penetration testing is significant. In reconnaissance, the tester gathers information about the test target in both active and passive ways. The goal of this step is to gather as much information as possible about the target to help plan the actual testing phase. [5]

Reconnaissance mainly focuses on getting information as an outsider of the target company [5]. Still, before starting this information gathering, it is important to have contact with the customer [10]. With the customer it is determined what the tester is testing, how wide and deep the testing area is, and how the results are reported. With these questions, it is agreed how many hosts are in scope, is the tester detecting vulnerabilities or trying to exploit them, and is there any contact person for situations when a critical vulnerability has been found for example. One really important thing is to settle the tester's authorization for testing [10]: is the tester allowed to copy the company website for themselves; are they allowed to do social engineering etc.

Passive reconnaissance is information gathering without interacting directly with the target. This phase could also be called open source intelligence, because we

gather information that is openly available [10]. It is possible to carry out passive reconnaissance in many ways: using Google, browsing company's webpage, reading manuals etc. It is possible to use all sources of free information at this point. [5]

Active reconnaissance means that information gathering is performed by interacting directly with the target [5]. This includes getting information from the Domain Name System (DNS), checking host names and Internet Protocol (IP) addresses and doing social engineering. Social engineering's goal is to exploit human weaknesses. A simple case of social engineering is to call a company employee, introduce oneself as an employee from an IT department and ask for the employee's username and password.

As a result of reconnaissance we should have agreed about details of penetration testing with the client, have gathered publicly available information [10] and have IP addresses to target the system [5].

2.2.2. Scanning

When we have enough information about the target system and the company, we can start scanning the hosts and network. Scanning can be divided into four sub-steps (shown in Figure 3): determine if the system is alive, scan ports, interrogate target further and scan system for vulnerabilities. The main purposes of these steps are to determine open ports, used services and identify known weaknesses. [5]

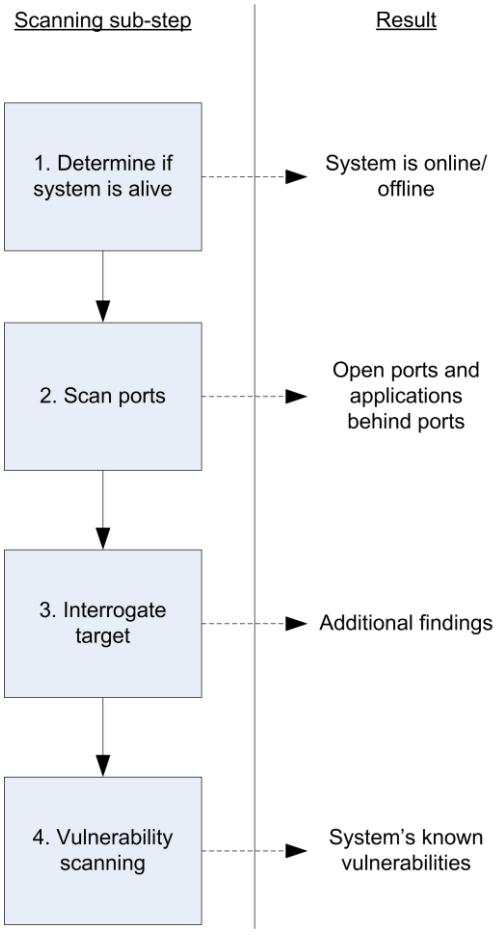

Figure 3. Scanning sub-steps

From the reconnaissance step we have the IP addresses of the system, but in this phase we should check with ping packets that the parts of the system are alive. It is worth noting that checking the hosts with ping packets is not a reliable way to determine whether the host is alive. For that reason, it is important to continue to the next sub-steps regardless of the result of this phase. [5] Still, we will get useful information on which hosts are alive for sure. If a host replies to a ping request, it is alive.

The second sub-step, scan ports, is to determine specific ports and applications that run on a specific port [5]. Port scanning is discussed in detail in Chapter 3.

Interrogating the target is quite similar to the previous steps. In this step, the tester goes deeper and deeper into the system to verify their earlier findings and make additional findings about the system [5]. When there is more information about the system, executing vulnerability scanning and exploitation will be easier.

Vulnerability scanning or identifying weaknesses is to check the system for known weaknesses. It is based on checking detected services and their versions from a vulnerability database, where services are linked to known vulnerabilities [10].

2.2.3. Exploitation

The exploitation phase is the process of gaining control over the system [5]. In the previous step, vulnerabilities have been detected, but all vulnerabilities are not necessarily exploitable and their criticality should be determined. For example, some exploits could reveal information, but the information might not be critical and does not give access to the whole system. This is an important step, because it is not effective to use resources to fix a vulnerability that does not provide an opportunity to exploit the system.

Exploitation is really a broad step, and there are a lot of activities, tools and options to perform this step [5]. Also, it is considered as the most interesting step in penetration testing [5, 10]. Because exploitation is a wide area, it is important to perform the previous steps properly: detailed information about the target system narrows down activities and options, which are possible or have to be done in the exploitation step. If reconnaissance and scanning is neglected, the exploitation will not be effective due to the wide range of options. For example, getting a user list in reconnaissance will greatly help password cracking.

The challenge of exploitation is that every system or target is unique. Systems vary on operating systems, services and processes for example. [5]

2.2.4. Post-exploitation

The post-exploitation phase can be defined as maintaining access [5] or information gathering on the exploited system [10]. It is not enough to determine that there is a vulnerability and that it is exploitable. For example, as a result of the exploitation phase we have remote access to one employee's e-mail. This does not necessarily tell us that attacker has access to critical information.

In the first decade of this century, many attackers used a "smash and grab" technique to gain access to a system only for the time necessary to download critical information. Nowadays it is more common that the attacker tries to have a long-term access to the target. This access is made possible with backdoors, so the attacker can

re-connect to the system anytime he/she wants. Another way is to use rootkits to get to the operating system to run hidden programs and processes. [5]

Post-exploitation also means making use of the vulnerabilities found. This can be done by escalating local privileges, gathering more information and finding ways to attack other systems. It is possible that the attacker or penetration tester does not get the desired level of privileges instantly. By gathering information it is possible to obtain passwords and other critical data to go deeper and deeper to the system. Also, it is possible to find files that contain sensitive information about the company, its employees and projects. Sometimes, the target machine can be a part of wider network or domain, and it can act as a gateway to other systems. This attack scenario is illustrated in Figure 4. [10]

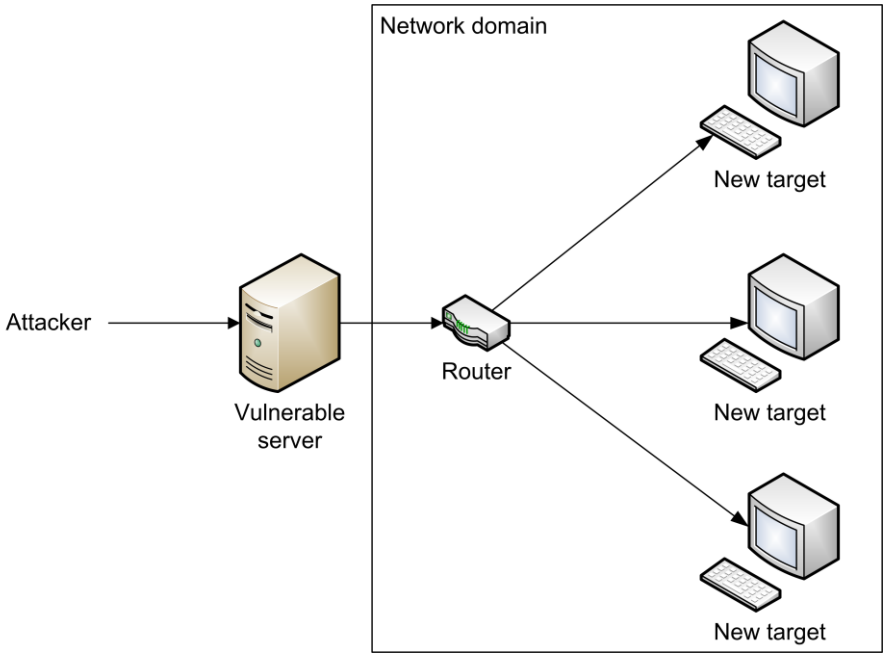

Figure 4. An attack scenario to access new targets through vulnerable server

2.2.5. Reporting

The main task of reporting is to conclude an analysis of the system, network, vulnerabilities and recommended actions so as to fix vulnerabilities [5]. Reporting is not only the last step where everything is put together, but data, text and information is gathered on every test step [4]. A penetration testing report is a critical task when there is a customer for whom testing is being performed. The report is sometimes the only evidence to the client of the results of the penetration testing.

The penetration testing report can be written at four levels: executive summary, walkthrough of how the testing was performed, detailed report and raw output. An executive summary is a brief overview of the findings of the testing process and is meant for board members and non-technical management. Therefore, an executive summary should be brief, and does not have technical details about testing and findings. The walkthrough of how the testing was performed is a document to show the steps in how the system is hacked or compromised successfully. The detailed report is a technical document of the findings of testing. It is used by IT managers

and network administrators to see what testing has covered and how to start fixing the vulnerabilities discovered. Lastly, the raw output from testing tools can be included in the testing report. Raw data may answer some of the client's questions that could be discussed by phone or in post-project meetings. [5]

2.3. External security databases

External security databases can be used in planning and executing security tests. There are databases for known vulnerabilities and attack patterns for example.

The Common Attack Pattern Enumeration and Classification (CAPEC) is an attack pattern database [13]. Attack patterns are methods and mechanisms that an attacker uses to gain access to a target machine. With this information, a software security tester can use the same steps and patterns in testing as a real world attacker would use. It is possible to do searches in the database by a keyword or CAPEC-ID number. One CAPEC entry contains a lot of information about the attack: summary of attack pattern, attack execution steps and preconditions, related weaknesses and attack patterns, payloads, and measurements of impact, severity and likelihood of exploit.

Common vulnerabilities and exposures (CVE) is a dictionary of publicly known security vulnerabilities that was launched in the year 2009 [14]. It was created because there were no standard for vulnerability names and metrics. Therefore it was difficult to determine whether two separate security testing tools refer to the same or different vulnerability. A CVE identifier has its own identifier number, description of the vulnerability and references. It is worth noting that CVE presents itself as dictionary, not a database. Therefore, the main purpose of the dictionary is to standardize the language and description of vulnerabilities.

When CVE is a dictionary of vulnerabilities and exposures, Common Weakness Enumeration (CWE) focuses on software weaknesses, and is targeted at software developers and security practitioners [15]. The dictionary of weaknesses is mainly focused on developing secure software. For example, CWE ID 256: "Plaintext Storage of Password" describes how storing passwords to a configuration file as plaintext weakens system and user security and how this issue should and should not be handled in implementation [16].

2.4. Limitations in security testing

Although penetration testing is a useful way to test software security, there are some limitations and issues that should be kept on mind. Many machines and networks are protected with a firewall and intrusion prevention systems (IPS). Also, penetration testing faces some legal issues, so it is not allowed to be used everywhere and in every situation. These issues can slow down the testing process and require authorization from the client to do penetration testing.

Intrusion prevention systems protect the target system to keep it safe. Still, in some cases it is reasonable to shut down the IPS. For example, the target system might have a protection method against port scanning, so it is not possible to send many network packets in a short time. This slows the scanning process significantly. Although port scanning could be slow with this system protection, it is possible to detect open ports and to try to exploit them. Therefore, IPS is only slowing the testing process, and all the vulnerabilities are not necessarily found.

Firewalls inspect network packets and decide whether the packet should be forwarded or dropped [17]. This might drop packets when scanning the host, so scanning results might indicate that host is not up or there are no open ports.

A penetration tester should always consider the legal and ethical issues when performing security tests. Many western countries have adopted “computer offences” or “hacking offences” in their law [18]. These offences include unauthorized access and interference with data, networks or computers for example. Even scanning hosts in a public network can be interpreted as interference in network. When there is a test target that belongs to a customer, it is important to agree what actions are allowed and what are not. Actual testing could be done in a closed security testing laboratory, so there is no access to public network. In that situation, it is ensured that tests will not affect to other computers or networks.

3. PORT SCANNING

Port scanning is a technique in which an attacker or software security tester tries to acquire information about the target host [19]. The attacker or tester tries to determine, for example, whether the host is alive, what services are running, what operating system the host uses and whether there are any exploitable vulnerabilities. This process can also be referred to as fingerprinting, which is defined as "the process of gathering all information available about computer systems in the network" [20]. The most important task for this process is to find open ports, the type of applications and operating system running on the target host.

3.1. Network ports

There are 65,536 (from 0 to 65535) different ports on every computer [21]. Depending on the service, these ports use Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). Network ports work as communication endpoints between hosts. They are not physical ports, but a logical construct that identifies a service or process.

Port numbers can be divided to three groups: well-known ports, registered ports and dynamic/private ports. Well-known ports are numbered from 0 to 1023. They are assigned by Internet Assigned Numbers Authority (IANA) and include commonly used ports such as Secure Shell (SSH), Hypertext transfer protocol (HTTP) and Hypertext transfer protocol secure (HTTPS). Registered ports are numbered from 1024 to 49151, and are registered to network services such as Internet Relay Chat (IRC) or Bit Torrent. Registered ports are also assigned by IANA. Dynamic/private ports are numbered from 49152 to 65535 and not assigned by IANA. They are used for local and dynamic use and it is not possible to use them as service identifiers. [21]

3.2. Fingerprinting methods

Fingerprinting has two methods to detect hosts and services in network: active and passive fingerprinting [20]. Additionally, hybrid fingerprinting is a combination of using these two methods. Another categorization for fingerprinting is to make a difference between operating system and application fingerprinting [22].

In active fingerprinting, network packets are sent to the target host and the replies are analysed. On the contrary, passive fingerprinting does not actively send packets to a host, but it captures the target host's network traffic and detects its operating system and used services this way. [20]

Active fingerprinting is an effective method to scan all network ports at once so as to detect whether a port is open, closed or filtered [20]. It is possible to get good and accurate results in a short time. Still, if the target host is using intrusion detection system (IDS), a scanner is likely to be detected and blocked [23].

The advantage of passive fingerprinting is that it does not generate extra traffic. Also, it is not detectable in IDS. A passive scanner is also eligible to do scanning offline by analysing previously captured packets. This method's disadvantage compared to active scanning is that it relies on packet capture. Therefore it is less accurate and complete than active fingerprinting. [23]

There are also different hybrid fingerprinting techniques [20, 24, 25] to combine the advantages of active and passive scanning. One implementation of the hybrid method is Hybrid Application Detection (HAD), which uses scanning tools Nmap, Amap and Ettercap to scan the target host [20]. This implementation is more accurate than using a single tool, especially in the detection application's version.

OS fingerprinting aims to detect the precise operating system the target host is using [22]. This can be used to monitor when systems should be updated, but OS fingerprinting has huge advantages in security testing. For example, when the precise operating system is detected, specific tests, attacks, probes etc. can be launched against the target host automatically according to OS [22, 26]. It is also worth noting that information on whether the operating system is Windows, Linux or Mac OS X is usually not enough, and the precise OS version should be available.

Application fingerprinting focuses on detecting ports, protocols, services and applications [20]. Each network port is linked to a specific service, and with proper fingerprinting it is also possible to get version of a used service [27].

3.3. Fingerprinting tools

As discussed in the Chapter 2.2, Port scanning and fingerprinting, it is possible to do active and passive fingerprinting. Active fingerprinting is beneficial in penetration testing, so it is possible to get scanning results quickly. Here it is presented three tools for active fingerprinting: Nmap, Xprobe2 and SPARTA. Nmap uses TCP and UDP packets for scanning, Xprobe2 is based on Internet Control Messaging Protocol (ICMP) along with TCP and UDP packets and SPARTA applies Nmap port scanning with a graphical user interface.

3.3.1. Nmap

Nmap is an open source tool for network discovery and fingerprinting. By sending raw IP packets, Nmap is able to scan the network for live hosts, what applications and services are running on those hosts, what operating system they are running and many other functions. It can be run on Windows, Linux and Mac OS X. Nmap is a command line tool, but it includes an advanced Graphical User Interface (GUI) and results viewer Zenmap; data transfer, redirection and the debugging tool Ncat; and a packet generation and response tool, Nping. It is possible to execute port scanning with control flags (also known as control bits) from the TCP packet header which is shown in Figure 5. Useful flags for port scanning with Nmap are synchronization (SYN), acknowledgement (ACK), reset (RST), urgent (URG), push (PSH) and finish (FIN). Also, the size of the window field is useful in port scanning. Additionally for TCP scanning options, Nmap provides an opportunity to scan UDP ports and IP protocols from the target host. [28]

bit	0-7	8-15	16-23	24-32
0	Source port		Destination port	
32	Sequence number			
64	Acknowledgement number			
96	Data offset / Reserved / Control flags (URG, ACK, PSH, RST, SYN, FIN)		Window	
128	Checksum		Urgent pointer	
160	Options			Padding
192	Data			

Figure 5. Control flags in TCP header

Port states

Nmap has six states that can be given to a port: open, closed, filtered, unfiltered, open|filtered and closed|filtered. [6 p.77-78]

When a port is recognized as open, there is an application behind the network port to actively accept TCP connections or UDP packets. Open ports are interesting in the security testing point of view, because they provide a venue to attack and exploit the system. Open ports can be protected with a TCP wrapper, but it still gives more opportunities for the attacker than a closed port.

Closed ports can receive Nmap probe packets, but there is no application receiving these packets. Recognizing these ports is useful for checking that the host is alive and detecting the operating system.

Filtered ports do not give so much information about the system, because probes do not reach the port. The port can be filtered by a firewall or router rules. Also some scanning methods cannot recognize that a port is closed. In this case, the port is labelled as filtered

It is possible for ACK scan to label ports as unfiltered. This means that port is accessible, but it is not possible to determine whether the port is open or closed. Other scanning methods might be useful in order to determine the port state.

Nmap can also place ports as state open|filtered and closed|filtered, when it is not able to recognize whether the port is open, closed or filtered. Closed|filtered case can occur only in Nmap idle scan. Open|filtered can be a result of UDP, IP protocol, FIN, NULL and Xmas scans.

TCP connect scan

TCP connect scan is a basic scan to discover open ports from a host [5]. It uses a three-way handshake, which is shown in Figure 6, to create and terminate the TCP connection. First a SYN package is sent to the target host to a specific port [29]. When the host replies with a SYN/ACK message, an ACK packet is sent to the target host. In this way, a connection has been established to two machines. If these steps are run successfully, the scanned port is open. After establishing a connection, the connection is terminated. With this functionality, TCP scan has only a little chance to flood and crash the target system, which makes scanning reliable.

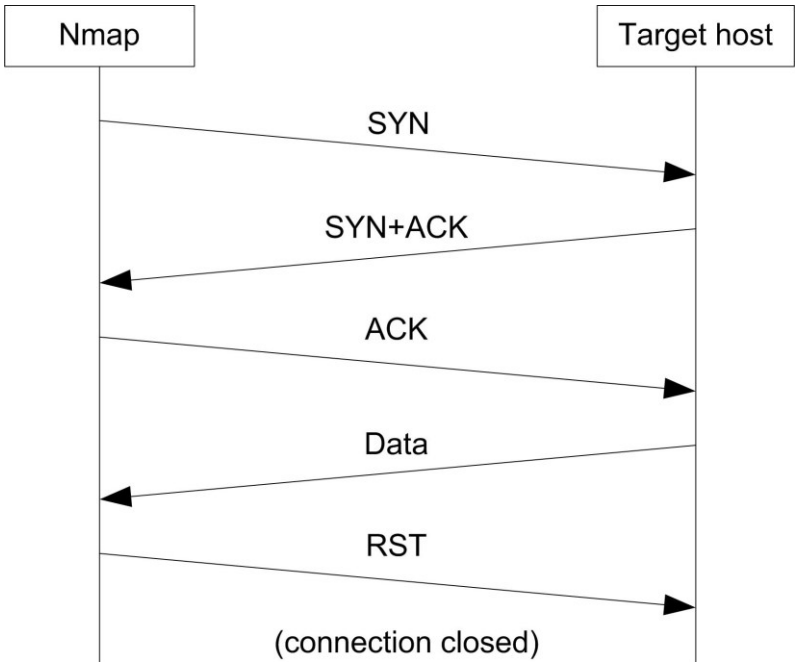

Figure 6. TCP scan to an open port

SYN scan

SYN scan is similar to TCP scan with a difference of one packet. It does not perform a three-way handshake fully. Figure 7 shows that when the host replies with SYN/ACK packet to SYN packet, SYN scan does not send an ACK message back to the host. It sends an RST package to tell the host to disregard previous packets and close the connection. Therefore, it is a little more efficient than TCP scan. SYN scan is also known as a "stealth scan", because the official connection is never 100% established. Therefore, not all logging applications get a log print, because they might require complete connection before starting recording. [5]

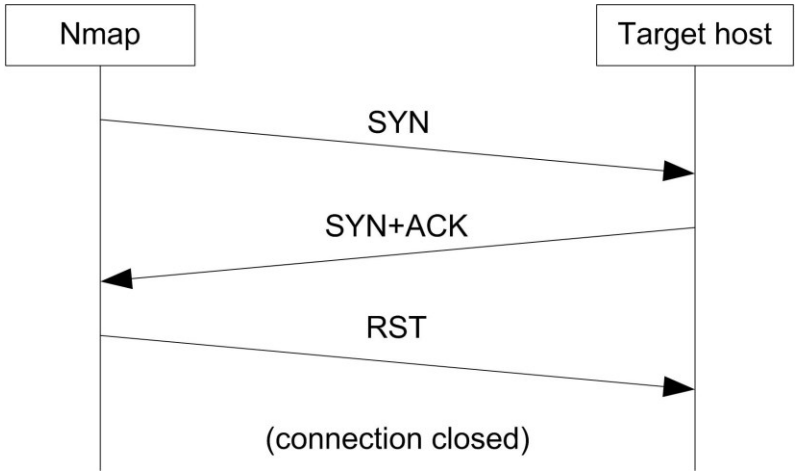

Figure 7. SYN scan to an open port.

SYN scan is able to label ports as open, closed or filtered. When the target host gives a SYN/ACK response to a SYN package, the port is open. In a closed port

situation, the scanner also receives a response. In this case, instead of SYN/ACK, the scanner receives an RST response. A port can be detected as filtered when the response is ICMP unreachable error, or when there is no response even after retransmission attempts. [6 p.97-99]

Custom, Xmas and Null scan

TCP protocol header has six control flags or bits that can be customized freely to carry out port scanning on the target host [28]. These flags are bit numbers 106-111 in the TCP header [29]. With a customised scan it is possible to carry out complex scanning by combining these control bits. By default, custom scan uses the same packet sending logic as SYN scan.

Xmas scan is quite an unusual scanning technique because it does not use SYN or ACK packets [5]. It uses FIN, PSH and URG packet flags. This functionality is implemented, because TCP RFC (request for comments) describes that, if the package does not have SYN, ACK or RST, a closed port should respond to this package with an RST packet [29]. If the port is open, the packet is ignored.

Null scan is an opposite method compared to Xmas scan. When Xmas scan violated traditional TCP communication by using unusual packet flags, null scan utilizes packet flags as empty. Here, an open port will not send responses to the package, but a closed port returns an RST packet. With null scan, it is possible to bypass filters from firewalls, for example. [5]

It is worth considering the Xmas and null scan's non-traditional nature in a couple of ways [5]. These scans are reliable only when the target host follows TCP RFC specification. Otherwise the host's replies to packets are not necessarily the same as anticipated. It is assumed that the target host will give an RST packet as a response when the port is closed and no response when the port is open. Still, it is not possible to determine that the port is truly open when there is no response, so Nmap labels the port as open|filtered if there is no response [6 p.107-108]. If the response is ICMP unreachable error, the port is labelled as filtered.

ACK scan

TCP ACK scan does not determine whether the port is open or closed. Still, it is useful to check firewall rulesets [30]. This scanning method labels ports only as filtered or unfiltered. If the target system is unfiltered, it gives a TCP RST response to both open and closed ports [6 p.113-115]. Therefore, it is useful to use ACK scan to acquire additional information about the system, not as an initial port scan.

Window scan

Window scan also uses ACK probes for scanning, but it checks the TCP Window value from a responded RST packet. This functionality is useful only for targets that have an implementation detail in the RST response: if the port is open, the RST packet has a non-zero window field. If the port is closed, the RST packet has a zero window field. Window scan is not always reliable, but can give additional information about the system. [6 p.115-116]

UDP scan

In addition to TCP header-based scans, it is also possible to use a UDP scan to discover User Datagram Protocol ports. When creating a connection with UDP, the three-way handshake is not used. Actually, a UDP connection is "connectionless", because there is no mechanism for a sender to be sure that sent packages have reached their destination. Therefore, UDP scanning is more difficult than TCP scanning, because the scanner does not get an instant response. UDP scanning is also a lot slower. [5]

It is rare that open UDP ports give a response to the sent packet [6 p.101-103]. When a port is closed or filtered, UDP ports usually give ICMP unreachable error. In some cases, there can be an answer from an open port. Then the port can be labelled as open. Still, a common result for a UDP port is open|filtered. The scanning method makes this deduction when there is no response from the target host even after retransmissions. Therefore, it is beneficial to use a version detection along with UDP scan [5]. Then Nmap uses nmap-service-probe to open, and open|filtered ports to detect the service and the service's version. With this method, it is possible to get more accurate scanning results.

Version detection

Also TCP scanning methods are possible with version detection. When TCP and UDP related scans' main task is to determine open ports, version detection can ascertain the service or application and its version. For example, Nmap might discover in SYN scan that TCP port 80 is open. Here Nmap checks from the Nmap-services database that there is probably an HTTP server behind port 80. This is not enough information for the penetration tester. With version scan, it is possible to detect what kind of web service is being used, for example. Although version detection along with port scanning is more useful, it is significantly slower. [28]

IP protocol scan

IP protocol scan is a little different compared to TCP, SYN, UDP, Xmas and null scan, because it does not scan TCP or UDP ports. It goes through IP protocol numbers and detects what protocols are in use by sending IP packet headers and goes through 8-bit IP protocol field [28]. There are 256 possible protocol numbers in total.

Responses and assigning protocol states are quite similar to UDP scan: if there is a response from a protocol, the target host supports that protocol. If there is no response, the protocol is open|filtered. If the response is ICMP unreachable error, the protocol state is set as closed or filtered.

NSE – Nmap scripting engine

Nmap has a scripting engine NSE that enables it to execute automated network tasks. There is a variety of scripts in Nmap itself, but users are also allowed to develop new scripts. NSE enhances basic Nmap functionalities with better network discovery, more sophisticated version detection, vulnerability detection, backdoor detection and vulnerability exploitation. Exploitation scripts are not included in Nmap in initial,

and Nmap developers do not consider creating exploits as a goal with Nmap development. Still, the user is able to create exploitation scripts for customized needs. [6 p.205]

There are nine different categories for Nmap scripts: *auth, default, discovery, external, intrusive, malware, safe, version* and *vuln*. Category *auth* contains scripts to acquire authentication credentials to the target system. *Default* is a default set of scripts. *Discovery* set is to get more information about the system, e.g. html title. *External* set uses third-party database or other network resources. A script is categorized as *intrusive* when it has high risk of causing a crash in the target system. *Malware* is to test malware and backdoor infections. *Safe* are contrary to intrusive scripts: safe scripts are not likely to cause a crash in the target. *Version* scripts are similar to version detection, but scripts can be more accurate when determining service version. *Vuln* scripts are developed to check for known vulnerabilities. [6 p.207-209]

3.3.2. Xprobe2

Xprobe2 is a command line scanning and fingerprinting tool and it includes 14 different modules for network scanning. The modules are shown in Table 1. It relies mostly on ICMP, but also implements TCP and UDP modules. Still, sending TCP and UDP packets is avoidable, so Xprobe2 is really quiet scanner and not easily detected by the target machine. Module 5, infogather:portscan, performs port scanning, but mostly Xprobe2's modules are beneficial on discovering whether the host is alive and fingerprinting the operating system. [23]

Table 1. Xprobe2 modules

Module	**Name**	**Description**
1	ping:icmp_ping	ICMP echo discovery module
2	ping:tcp_ping	TCP-based ping discovery module
3	ping:udp_ping	UDP-based ping discovery module
4	infogather:ttl_calc	TCP and UDP based TTL distance calculator
5	infogather:portscan	TCP and UDP Port Scanner
6	fingerprint:icmp_echo	ICMP Echo request fingerprinting module
7	fingerprint:icmp_tstamp	ICMP Timestamp fingerprinting module
8	fingerprint:icmp_amask	ICMP Address mask fingerprinting module
9	fingerprint:icmp_info	ICMP Information request fingerprinting module
10	fingerprint:icmp_port_ unreach	ICMP port unreachable fingerprinting module
11	fingerprint:tcp_hshake	TCP handshake fingerprinting module
12	fingerprint:tcp_rst	TCP RST fingerprinting module
13	fingerprint:smb	SMB fingerprinting module
14	fingerprint:snmp	SNMPv2c fingerprinting module

ICMP has 11 different message types, which have a different structure on the ICMP field in the IP packet header [31]. Still, the first 32 bits of the field are structured as shown in Figure 8. The first 32 bits define the type of message, code that relates to the type of message and the checksum of the first 16 bits. Xprobe2 utilizes five types of ICMP messages: echo, timestamp, address mask, information request and port unreachable. Usage of these messages is simple: Xprobe2 sends an ICMP package and analyses the reply from the target host.

bit	0-7	8-15	16-23	24-32
0	Type	Code	Checksum	
32	Type specific fields			
...	...			

Figure 8. ICMP field structure

As an example of operating system fingerprinting, Xprobe2 uses ICMP echo request [23]. With echo request, it analyses three pieces of information: Explicit Connection Target (ECT) option, time to live (TTL) and flag. In Listing 1 is shown packet data from ICMP echo request sent from a Kali Linux 4.0 host and the reply from a Windows 7 host. From this packet data, it is possible to determine that the target host does not follow ICMP RFC, because the flag in reply is changed from DF to none. If the target host follows ICMP RFC, the same flag DF would have been sent in the ICMP echo reply message. Also it is worth noting that the target host does not support ECT and TTL is not decreased to the value of 64 so both hosts are in the same broadcast domain.

Listing 1. ICMP echo and reply

```
1  09:52:41.611446 IP (tos 0x6,ECT(0), ttl 64, id 3993, offset 0, flags [DF],
   proto ICMP (1), length 84)
   kali > 192.168.182.2: ICMP echo request, id 59700, seq 1, length 64
2  09:52:41.611534 IP (tos 0x0, ttl 128, id 34309, offset 0, flags [none], proto
   ICMP (1), length 84)
   192.168.182.2 > kali: ICMP echo reply, id 59700, seq 1, length 64
```

For port scanning, Xprobe2 uses TCP packets with the SYN flag to determine port states. Functionality of the scanning is the same as in Nmap's SYN scan: first Xprobe2 sends a TCP packet with the SYN flag; after that a response is sent from the target host, and finally a three-way handshake is not fully done as Xprobe2 sends an RST message back to the target host. This functionality to open port is illustrated in Listing 2.

Listing 2. Port scanning with Xprobe2 on open SSH port

```
1  10:11:41.828857 IP kali.38723 > scanme.nmap.org.ssh: Flags [S], seq
   1672663691, win 5840, length 0
2  10:11:42.012773 IP scanme.nmap.org.ssh > kali.38723: Flags [S.], seq
   755243184, ack 1672663692, win 64240, options [mss 1460], length 0
3  10:11:42.012798 IP kali.38723 > scanme.nmap.org.ssh: Flags [R], seq
   1672663692, win 0, length 0
```

Xprobe2 is no longer under development and the latest version was released in July 2005. Still, Xprobe2 can be a useful tool to detect older operating systems especially when there are no open ports. [23]

3.3.3. SPARTA

SPARTA is a network infrastructure penetration testing tool that uses Nmap so as to map network hosts and scan for open ports. It has a graphical user interface (shown in Figure 9) for showing hosts, scanning results and test results. The scanning can be done actively in a target network or Nmap Extensible Markup Language (XML) file can be imported. SPARTA uses five stages model to scan target network starting from common ports in the first stage and ending in dynamic and uncommon ports on the fifth stage. During scanning, SPARTA also launches automated tests on different ports. For example, when it recognizes that HTTP port 80 is open, it automatically runs the test tool Nikto. [32]

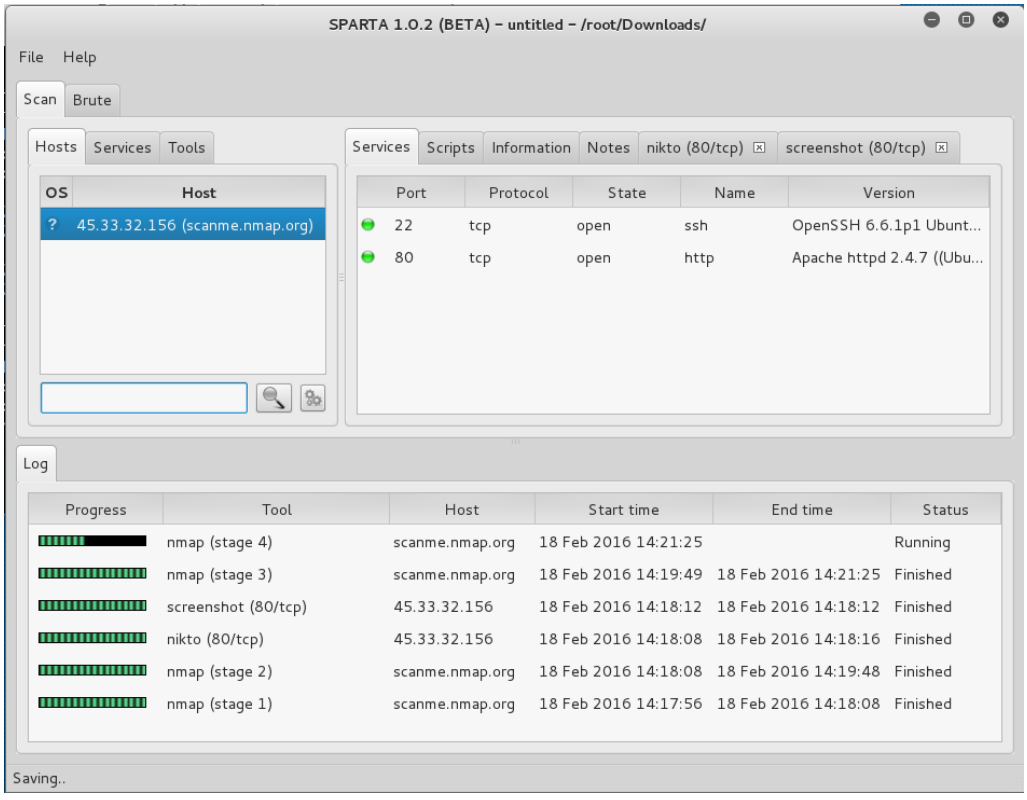

Figure 9. SPARTA user interface

For all these scans, SPARTA uses Nmap's SYN scan (default option), and scans are accompanied with version detection (option –sV) and a little faster timing template than default is used (option –T4). Also UDP ports are scanned with the base of SYN scan (option –sSU). The target can be specified as a single IP address or IP address range.

When all five stages have been run, SPARTA has scanned all TCP ports and six UDP ports along with version and operating system detection. Detailed information about the scans can be extracted from XML output files.

Nmap command in the first scanning stage is “nmap -T4 -sV -sSU -p T:80,443 <target>”. In the first stage, SPARTA scans two common TCP ports, numbers 80 (HTTP) and 443 (HTTPS).

In the second stage, SPARTA takes more TCP ports into scanning and also checks four UDP ports. Scanning is also accompanied with OS detection and cancelling DNS resolution. The full Nmap command is “nmap -T4 -sV -n -sSU -O -p T:25,135,137,139,445,1433,3306,5432,U:137,161,162,1434 <target>”.

In the third stage, SPARTA takes eight TCP and two UDP ports into scanning. The full command for scan is “nmap -T4 -sV -n -sSU -p T:23,21,22,110,111,2049,3389,8080, U:500,5060 <target>”.

After the first three stages, SPARTA scans all the missing ports from the last steps until to port number 29999. To this command, option –Pn is added to disable host discovery, and Nmap assumes that all target hosts are online. The full command for the stage four scan is “nmap -Pn -T4 -sV -n -sSU -p T:0-20,24,26-79,81-109,112-134,136,138,140-442,444,446-1432,1434-2048,2050-3305,3307-3388,3390-5431,5433-8079,8081-29999 <target>”.

At the last stage, all remaining TCP ports are scanned with command “nmap -Pn -T4 -sV -n -sSU -p T:30000-65535 <target>”. This stage includes scanning ports from number 30000 to number 65535.

3.4. Analysis of fingerprinting tools

Fingerprinting tools can be compared with eight different terms that describe tools’ functionalities. This comparison is shown in Table 2. User interface indicates that the tool uses graphical user interface or is run on command line. Reporting is the output, how the scanning and testing results can be extracted from the tool. Maintenance means the development activity of the tool. Port scanning, OS fingerprinting, application fingerprinting and security test functionality indicate that functionality is included in the test tool. Finally, flexibility means how many different options there are to execute listed functionalities.

Xprobe2 focuses on operating system fingerprinting and lacks functionality in port scanning. Therefore it cannot be considered as a versatile and useful in penetration testing. Xprobe2 has an advantage in operating system fingerprinting with ICMP packets over Nmap and SPARTA. Still, the last version of Xprobe2 was released in July 2005, so the tool is distinctly outdated, as fingerprinting is possible for operating systems that have been released before year 2005.

For these three tools, Nmap has the best cover of listed functionalities and options to run port scanning. There are over 100 different scanning methods that utilize TCP packet. The developer of Nmap, Gordon Lyon, is active in improving all aspects of Nmap: the program in general, scripting engine scripts and documentation. Also, Nmap has an active community that develop new scripts and give ideas for enhancing the tool.

SPARTA can be considered as a corresponding implementation to the one developed on this thesis. Still, SPARTA lacks some functionality that is essential to the penetration tester’s work. Although it runs some tests automatically and it is possible to implement new testing scripts, it does not provide good reporting of the

test results. Also, SPARTA is developed to perform scanning and testing on wider networks than one target host. Nor does it get comprehensive scanning results, because it only uses SYN scan to determine the TCP ports' state.

In this thesis, Nmap is going to be used for port scanning. Compared to Xprobe2 and SPARTA, it has the most versatile approach to using different kinds of scanning methods. Nmap also covers UDP port and IP protocol scanning in addition to TCP port scan. Based on these scans, it can also detect service with version behind an open port. Nmap has over 90% accuracy in detecting service and over 50% accuracy when detecting version [20].

Table 2. Comparison of fingerprinting tools

	Nmap	**Xprobe2**	**SPARTA**
User interface	Command line, GUI available	Command line	GUI
Reporting	Plain text, XML	XML	SPARTA project format
Maintenance	Active community and developer	Last version released on 2005	Beta development ongoing
Port scanning functionality	Many different scanning options	Scanning limited to SYN packets	Scanning limited to SYN packets
OS fingerprinting functionality	Included	Good with ICMP packets, but outdated.	Included
Application fingerprinting functionality	Included	Not included	Included
Security test functionality	Included with Nmap scripting engine	Not included	Included with external tools and scripts
Flexibility	Possible to use many different scanning techniques and develop own NSE scripts	No flexibility with scanning methods and automated tests not possible.	Poor flexibility with scanning methods. Possible to develop test scripts with external tools.

4. AUTOMATED TESTING IMPLEMENTATION

In black-box penetration testing, port scanning is a task that has to be done every time before testing. Also some tests are designed to give information and results of known vulnerabilities in certain services. It is reasonable to automate these tasks to make the penetration tester's work easier and save time on running routine commands manually. As a result of scanning and tests, there might be quite a lot of data so reporting of results is an essential part of scanning and testing implementation. Results are exported to a clear report which can be used in penetration testing report.

For automated port scanning, it is important to get comprehensive scanning results. After the tester has run the scanning, there should be good amount of information available to start actual penetration testing. Many different scanning methods can be used to get information from the target host. The goal is to gather information not only from TCP and UDP ports, but also IP protocols used.

This implementation is designed to scan a single network host, not wider networks. When only one host is scanned and tested, it is possible to use more time to the target and get better results. In wider networks, it is possible that scanning and testing might take several days which is not convenient to support testing.

The architecture of the implementation includes two modules and one library as shown in Figure 10. The scanning module executes port scanning for TCP ports, UDP ports and IP protocols with Nmap. The module also scans for services behind open ports and runs operating system detection. The testing module runs automated security tests, which are included in the Test library. The library consists of test sets, and these sets can be executed according to the services that have been found from the target host. Both scanning and testing modules create a report as plain text to have human readable format and XML, so results can be processed for example to an HTML page.

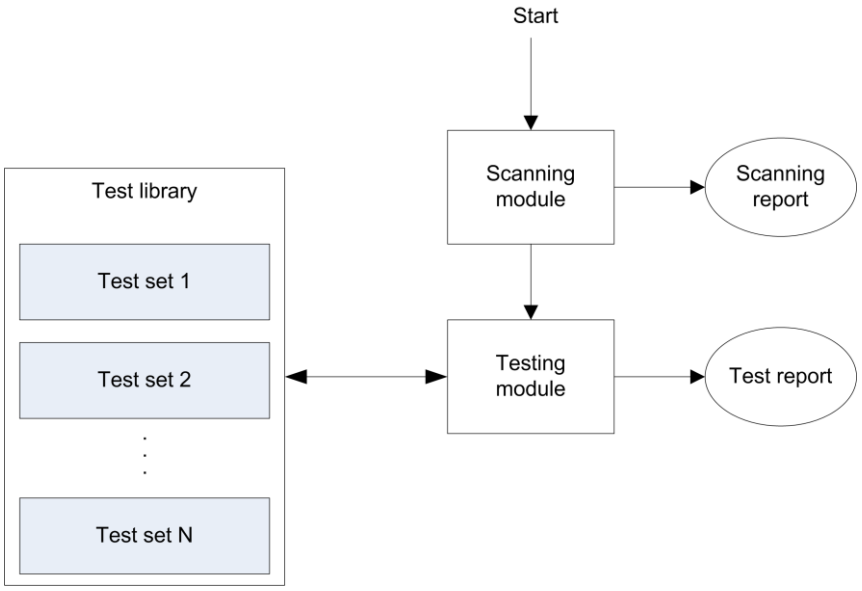

Figure 10. Architecture of automated testing implementation

Comprehensive scanning and testing can be time-consuming. To support the penetration tester's work, it is important to get some test results in a short time. Timing can be divided into three categories to support the tester's schedule:

- Quick scan: the tester gets a report in a couple of minutes, so actual testing can be started quickly.
- Default scan: the tester gets a report in one or two hours. In this case, scanning can be run during a lunch break or meeting.
- Comprehensive scan: the tester gets a full report with no restrictions in scanning time. This scan can be run outside working hours.

With these scanning schedules, the tester is able to choose an appropriate method to start testing. Sometimes it is not even necessary to run a comprehensive scan or automated tests to start testing. These three categories have the following scanning coverage:

- Quick scan: 100 most common TCP ports. Version detection, operating system detection and security tests are not run.
- Default scan: 1,000 most common TCP ports, 1,000 most common UDP ports and IP protocols with version and operating system detection and automated tests.
- Comprehensive scan: all TCP ports, 1,000 most common UDP ports and IP protocols with version and operating system detection and automated tests.

With default and comprehensive scan, there is also an option that automated tests are not run.

4.1. Scanning TCP ports

The scanning process of TCP ports can be divided into four steps shown in Figure 11. The first step is to run port scanning with different methods; the second step is to make a deduction of the single port's state; the third step is to execute version detection to open ports and run operating system detection; and finally, the fourth step is to create testing report.

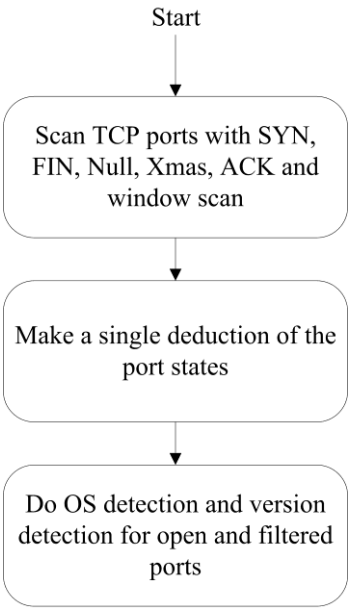

Figure 11. Steps of TCP scanning script

First, Nmap's different port scanning methods are used to get information on ports. The methods used are SYN, FIN, Null, Xmas, ACK and window scan. With these scanning methods, it is possible to acquire comprehensive information about the port and additional information on the port's behaviour in different situations. In addition to port information, scanning methods retrieve other information from the host: media access control (MAC) address, IPv4 address, IPv6 address and the vendor of the network device.

The second step is to make single deduction of the port's state. Results from all scanning methods should be combined and shown in the scanning report, but also single deduction of the port state should be derived. This implementation uses six different scanning methods that mostly work differently. Only FIN, Xmas and Null scans have the same behaviour as each other,

To make a single deduction of the port state, it is important to analyse responses for each port scanning method. Scanning methods used, possible responses and port state assignments are shown in Table 3. It is possible to get port state open, filtered, unfiltered, open|filtered or closed as a port state assignment when using TCP scanning methods. Still, there is a variation in what kind of results it is possible to get from a certain method. For example, ACK scan will not assign a port as "open" or "closed". It uses the assignments "filtered" and "unfiltered", because response to ACK message is RST both on open and closed ports when the system is not filtered with a firewall or other IDS or IPS.

Table 3. Scanning functionality of TCP scans

Scan method	**Open**	**Filtered**	**Unfiltered**	**Open| filtered**	**Closed**
SYN scan	SYN/ACK response	No response or ICMP unreachable			RST response
FIN scan Xmas scan Null scan		ICMP unreachable		No response	RST response
ACK scan		No response or ICMP unreachable	RST response		
Window scan	RST response with non-zero window field	No response or ICMP unreachable			RST response with zero window field

These responses and assignments can be divided into trusted and non-trusted assignments. A trusted assignment is a port state that can be used as a final deduction of the port's state. On the other hand, if a port state assignment to a certain response is considered as not to be used in the final deduction, the assignment is non-trusted.

Analysis can be started by ruling out untrusted assignment of the port state. When the target host does not give any response to a TCP packet, the situation is

considered untrusted. A not responding scan does not give a lot of information of the target's behaviour, so the correct port state has to be derived from other scanning methods. Also, an RST response can be considered as untrusted in general, because it gives closed or unfiltered assignment. In this case, other scanning options can give better results. All scanning methods have a no response option to give untrusted port state assignment. SYN, FIN, Xmas and Null scan can give an untrusted result with RST response. Also, ACK and window scan utilize a RST response in a port state assignment, but the functionality is different with these methods.

Port state assignment can be handled as trusted when the port has responded and the response indicates open or filtered behaviour on the scanned port. When open or filtered state is assigned based on SYN/ACK or ICMP unreachable responses, the assignment can be considered to be trusted. In this case, the scanned port has been active to the sent message, and it indicates that the port is open or filtered. This gives one attacking interface to the penetration tester. SYN, FIN, Xmas and Null scan uses SYN/ACK or ICMP unreachable response to make a trusted port state assignment. Again, ACK and window scans have a different logic on functionality, so they cannot be considered to be trusted assignments.

As discussed before, ACK scan works differently from other scanning methods: it determines firewall or other IPS/IDS's functionality. Therefore all results should be considered as untrusted when making a deduction of a port's state. Still, ACK scan should be run and results should be reported.

Window scan has clear functionality for RST responses: if there is a zero window field in the response, the port is closed. If the response has a non-zero window field, the port is open. Still, depending on the test target, the target can send an RST response with a zero window field, although the port is actually open. Interesting results in windows scan are anomalies in results that differentiate from the other results. For example, if the target host has 65,532 open ports and three closed ports, the three closed ports should be examined more closely. It is really unusual that the target host would have more open ports than closed ones. Therefore, it is reasonable to use a smaller group of results in the deduction.

Summary of trusted and untrusted assignments are shown in Figure 12. There are five trusted assignments that can be used to deduct a single state for a port. Trusted options for port states are open and filtered. If the response from SYN scan is SYN/ACK or ICMP unreachable message, it should be considered as a correct port state. When SYN scan assigns port as open, it is reliable that the port really is open. If SYN scan does not give trusted assignment, then a trusted result from FIN, Xmas and Null scan can be used. Results from window scan can be used, if other scans do not give a trusted assignment. Finally, if there are no trusted assignments for a port, the port can be deducted as filtered, open|filtered or closed based on untrusted assignments.

The last step in scanning TCP ports is version and operating system detection. Nmap has built-in detection systems for this. Version detection is run to open and filtered ports so as to determine the service and number of service version. This is an important step, because without version detection, Nmap reads the port's service from Nmap's service database. In that case, Nmap assumes that the port is used to the service that it is assigned to. Still, it is possible that e.g. web service is not running behind traditional HTTP, HTTPS or HTTP-proxy ports. Therefore assuming the service behind a port without version detection can be almost the same as guessing especially with registered and private ports. Sometimes, Nmap is not able to

determine the version for the target host's response. In this case, the response is printed as plain text so the user is able to interpret the response.

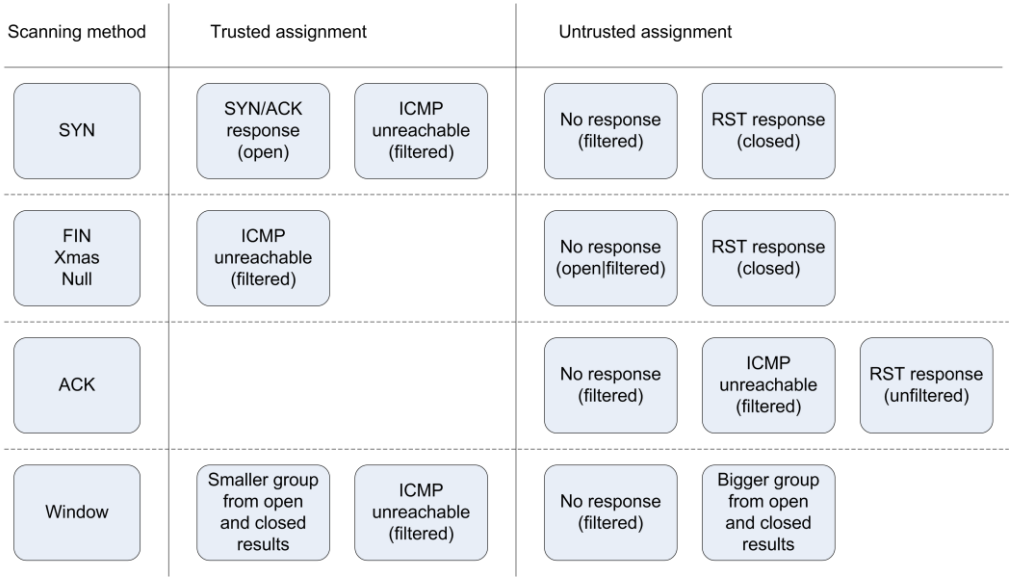

Figure 12. Summary of trusted and untrusted assignments

Operating system detection also requires closed ports in order to determine the system running on the target host. Therefore, at least three random well-known closed ports should be added so as to carry out operating system detection.

4.2. Scanning UDP ports

Scanning of UDP ports has only two steps to perform port scanning. UDP scan has fewer steps than TCP scan to process, because Nmap provides only one method to run UDP scan. The two steps, scanning UDP ports and doing the version scan, are shown in Figure 13.

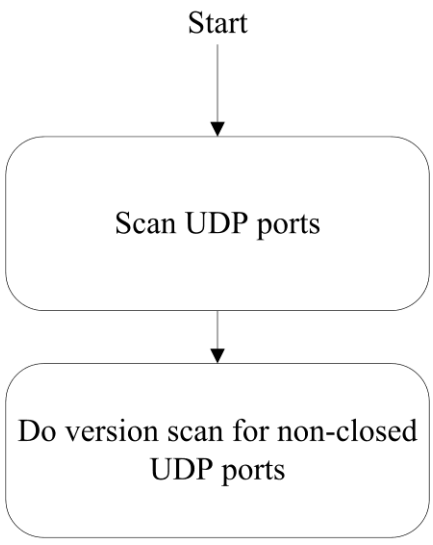

Figure 13. Steps of UDP scanning script

Although scanning UDP ports is difficult at the packet level, Nmap provides only one method to run the scanning. Responses to Nmap's sent packages and port state assignments are shown in Table 4. When the target host gives any UDP response to the sent UDP packet, Nmap labels that port open. There can also be an ICMP unreachable error. In that case, scanned port is assigned as filtered or closed depending on ICMP error's code. When there is no response to the UDP packet, the port is assigned as open|filtered. That is the most common state for a UDP port, because the ports usually do not respond to the sent messages.

Table 4.Scanning functionality of UDP scan

Scan method	**Open**	**Filtered**	**Unfiltered**	**Open| filtered**	**Closed**
UDP scan	Any UDP response	ICMP unreachable error (type 3, code 1, 2, 9, 10 or 13)		No response	ICMP unreachable error (type 3, code 3)

Also, UDP scan utilizes version detection to scan for services behind non-closed ports. The purpose and functionality of version detection is the same as in TCP ports version detection: without version scanning, the service behind a non-closed port is read from a database. With version detection it is possible to get a more accurate assignment for the service behind a port.

4.3. Scanning IP protocols

Scanning of IP protocols has quite similar steps and protocol state assignments to scanning of UDP ports. Nmap has a built-in feature to cycle through IP protocols to check whether the protocol is open, filtered, open|filtered or closed. The scanning process contains only one step as shown in Figure 14.

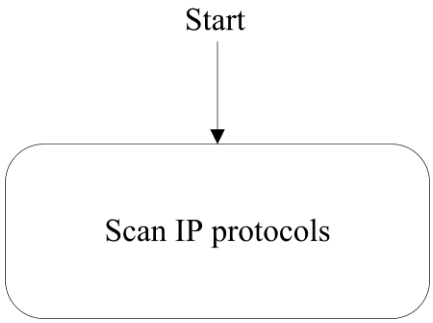

Figure 14. Steps of IP protocol scanning

Behaviour in port state assignments is almost the same as in UDP protocol scanning. The assignments are shown in Table 5. One difference with UDP scanning behaviour is with ICMP unreachable error: when UDP scanning makes closed assignment with code 3, IP protocol scan makes closed assignment with code 2. Any

other response assigns the port as open. Open|filtered assignment does not give much information on target protocol, because the target does not response in that case.

Table 5.Scanning functionality of IP protocol scan

Scan method	Open	Filtered	Unfiltered	Open| filtered	Closed
IP protocol scan	Any response in any protocol	ICMP unreachable error (type 3, code 1, 3, 9, 10 or 13)		No response	ICMP unreachable error (type 3, code 2)

4.4. Scanning report

An essential part of the scanning process is to report the scanning results. After scanning, there are quite a lot of data that has been gathered from the target host in the port scanning process: scan runtime, MAC address, IPv4 address, IPv6 address, vendor, scanning results of TCP ports with service and version number, scanning results of UDP ports with service and version number, scanning results of IP protocols, detection of operating system and possibly some plain text responses from version scanning. This information has to be reported clearly so human can easily understand the contents and the report should be brief to present only necessary data. The main idea of the report is to support the penetration tester's work.

Reported data can be divided into three categories: general info, basic results and additional results. The categories and their contents are shown in Figure 15. General info presents the information that is not related to the ports and IP protocols that have been scanned. This information includes host's addresses, vendor, detected operating system and other data relating to the scanning process, e.g. scanning time. Basic results show brief data of the target system so the reader is able to get a good knowledge of open ports in the system. These results include a count of scanned ports and port state assignments with different scanning methods, non-closed TCP and UDP ports with services, versions and IP protocols used and suggestions of test tools that can be used in penetration testing. The suggestions give instant guidance to the penetration tester to start testing. Additional results show more detailed information about scanning results. Port state assignments with different scanning methods, a deduction table which scanning method was used to deduce state for a non-closed port and responses from version detection with an unidentified version is shown to offer more data to analyse the system's behaviour.

4.5. Running security tests

After open ports are discovered in the scanning module, it is relevant to run security tests on the services that are running on those ports. There are large number of security testing tools available both commercial and open source. Running all possible tools to all services found would not be reasonable or even feasible, because running time would be really long. For example fuzzing, which is "a kind of method by offering unexpected input to the target system and monitor abnormal results to

discover software vulnerabilities" [33], can be run with massive dictionaries or randomly generated inputs which can be run infinitely. Therefore, it is important to choose tools and scripts to security test automation carefully.

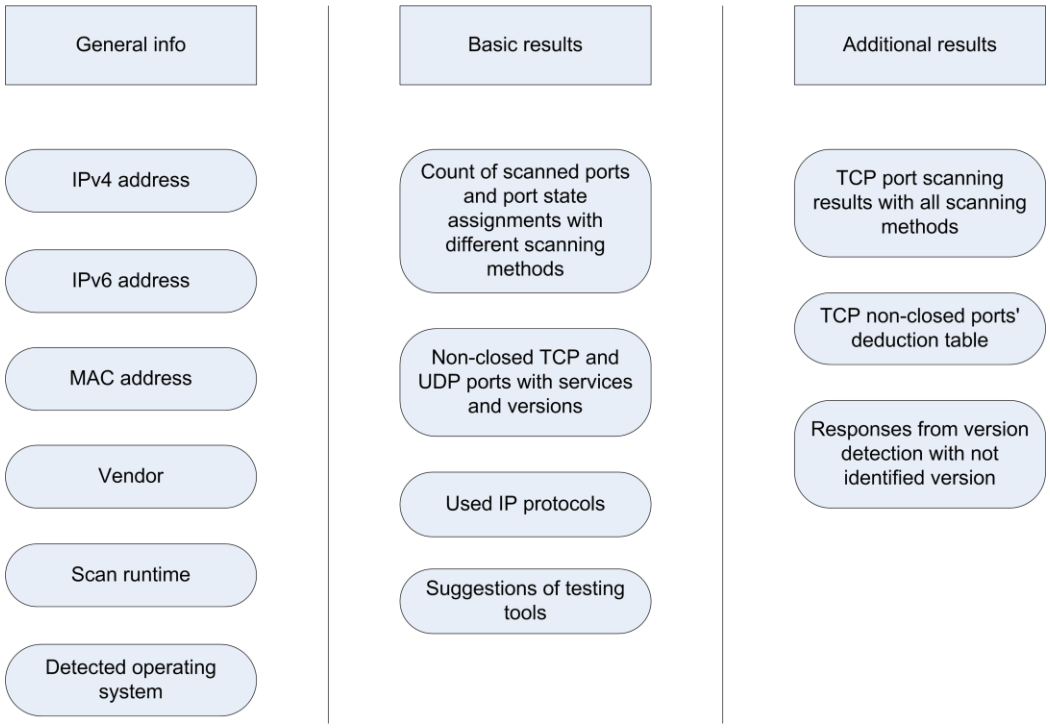

Figure 15. Categorization of scanning results

The test module consists of a core test module and test library as shown in Figure 16. The test library can include tests for many different services, but here only HTTP testing with Nikto [34] and general security testing with Nmap scripting engine are shown. Both test tools provide not only found vulnerabilities or security flaws, but also a great deal of "information only" data from the test target. This information is beneficial for the security tester to learn more about the target machine and plan other security tests.

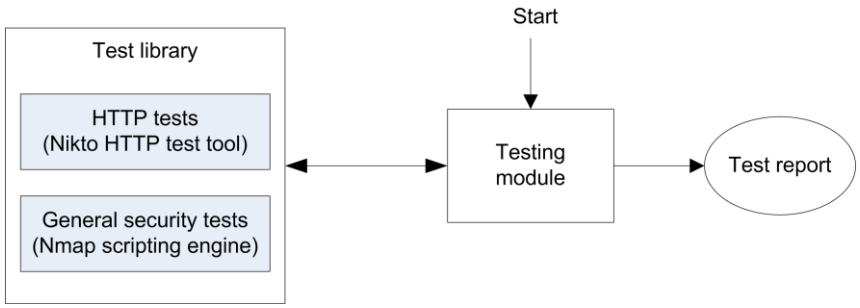

Figure 16. Architecture of implemented test module

Nmap scripting engine is a general tool for vulnerability scanning and gathering information from a target system. There are 516 scripts to run. It is not appropriate to run all the scripts on every host. The scripts do not have labels according to the target

port or service. Therefore, it is a little difficult to choose appropriate scripts to match the target host. In this implementation, the parsing is simplified to match the script's name: if a Simple Network Management Protocol (SNMP) service is found from the target host, the script that has "snmp" in its name is executed. In this case, NSE will execute 12 scripts which are listed in Listing 3. In addition to service related scripts, NSE's discovery category is run to gather more information from the target host.

Listing 3. Nmap scripting engine scripts for SNMP service

```
1  snmp-brute.nse
2  snmp-hh3c-logins.nse
3  snmp-info.nse
4  snmp-interfaces.nse
5  snmp-ios-config.nse
6  snmp-netstat.nse
7  snmp-processes.nse
8  snmp-sysdescr.nse
9  snmp-win32-services.nse
10 snmp-win32-shares.nse
11 snmp-win32-software.nse
12 snmp-win32-users.nse
```

Nikto is a web application and server scanning tool to check for the server's configuration, versions and potentially dangerous files, and it is implemented to the Test library's HTTP tests. Nikto also provides informational output so that the penetration tester is able to learn more about the target host, which is beneficial in planning further test cases. For example, Nikto might report that there is a folder called *auth* in the target host's web application. Nikto does not provide information on whether there are vulnerabilities or not, but suggests the penetration tester check that folder and run tests on it.

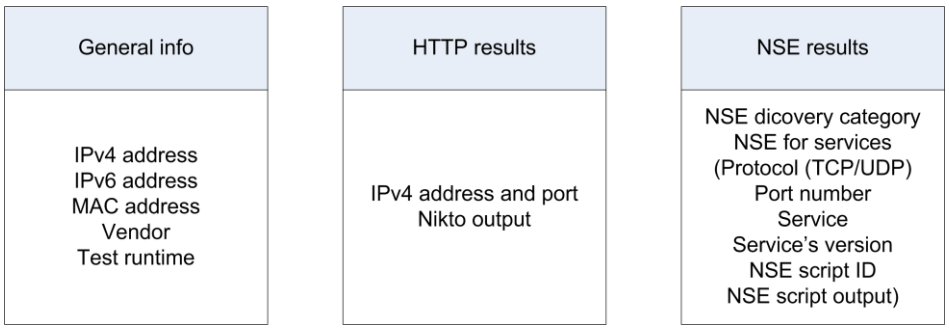

Figure 17. Categorization of testing results

The test module creates test result reports in two formats: plain text and XML. The results are raw data from test tools so there is no filtering for data. In this way, all data discovered is shown. Reported data has general info about the host and the rest of the results are divided according to the Test library's test sets. In this case, the categories are general results, HTTP results and NSE results as shown in Figure 17. General results include general information about the target host: IPv4 address, IPv6 address, MAC address, vendor and test runtime. In HTTP results, there is an IPv4

address and port to specify the test target in detail. It is possible that one host has more than one web service running in different ports. NSE results include results for NSE discovery category and NSE for services. This last consists of protocol, port number, service running behind the port, service's version, NSE script ID and NSE script output.

5. TESTING OF THE IMPLEMENTATION

Testing of the scanning and security testing implementation is carried out in two scenarios: the first is towards one target to illustrate the implementation's functionality and handling of test results. The second scenario is to execute scanning and testing on other network devices so as to verify that the implementation is deployable to various test targets. The first scenario is executed with Network Edge Device (NED) which is developed under SECURED project. In the first scenario, NED is the system under test. The second scenario is performed with two unknown test targets to do black-box security testing in an extreme situation: there is no information about the SUT other than the IP address to the target. The scanning and testing implementation is also deployed inside to the NED in order to scan the NED's inner components.

5.1. Network edge device (NED) as system under test

Nowadays, there are more and more personal network devices in use [35]. Mobile phones, tablets, laptops, smart TVs, wearable technology and other network devices give malicious users new interfaces to exploit vulnerabilities. This security threat builds up from the number of devices and software, especially when the same user has many different devices. It is challenging to keep all the devices secured, and all not users have the skills and knowledge to protect their devices from security threats. Also, devices may have limited resources to run security applications.

In the SECURED project, a proposal to solve the presented problem is to move security protection from the user device to the Network Edge Device (NED). The main idea is that security protection is not device- or network-based. It is user-specific, so it does not matter which network or device the user uses to get security protection with the SECURED system. The NED can contain anti-virus, anti-malware, ad blocking, parental control and single web page blocking, for example. [36]

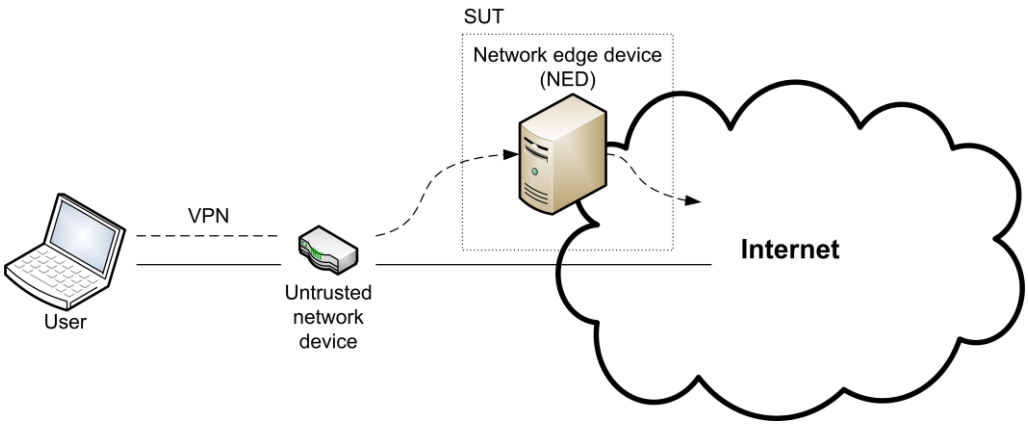

Figure 18. Remote NED scenario in SECURED system

In this thesis, NED is going to be the system under test with automated testing implementation. NED's architecture can be configured in many ways, but here remote NED scenario is used as shown in Figure 18 [35]. In this scenario, the user

connects remotely to NED with a virtual private network (VPN) and all networking to the internet is passed through NED. Still, in this testing scenario the VPN connection is not established. This attack scenario represents a situation where the attacker tries to access the NED without a VPN connection.

Now in the year 2016, SECURED project is still ongoing and testing is going to be performed to NED's basic architecture. It is notable that the environment in this case is constructed for integration testing and final hardware or software of NED is not used. Some results from these tests can be related to the integration test environment setup itself, not to the NED functionality. When these aspects are taken into account, it is noted that all test results do not apply to the finished version of NED. Still, NED is a good target to test the scanning and testing implementation.

5.1.1. Quick scan

There are 100 TCP ports scanned with quick scan and a full report of this scanning is shown in Appendix 1. This is the method to get results in a short time so there is some information about the target host in a short time. The scanning took one minute and 11 seconds.

The results of quick scan are quite minimal, but they give a first glimpse of the system under test. In this case, there are two open and one filtered port discovered as shown in Table 6. There are three interfaces where the penetration tester can start to plan attack scenarios and testing methods. The scanning and testing implementation suggests using the Hydra [37] tool to perform a password attack on the SSH port. With these results and suggestions, it is possible to start security testing in a less than two minutes, even though there is no prior knowledge of the target.

Table 6. Port scanning results with quick scan

TCP port	**Service**	**Deducted state**	**Scanning method**	**Suggested testing tools**
22	ssh	open	syn	Hydra
111	rpcbind	open	syn	
514	shell	filtered	syn	

In Chapter 4.1 Scanning TCP ports the fact that this implementation assigns port scanning results to trusted and untrusted assignment was discussed. The scanning is run with six different TCP port scanning methods. Results from these scans are shown in Table 7. There are no results shown from NULL, FIN, Xmas and ACK scan, because they did not provide trusted scanning results. Only SYN and Window scan gave trusted assignments. It is worth noting that for port 514 SYN scan gives the assignment *filtered* and window scan labels that port as *open*. The final state for the port is *filtered,* because SYN scan is considered to be more reliable scanning method than window scan. Also, it is worth noting that, with SYN scan, port 514 is an anomaly, when it acts the same as other ports in window scan.

Table 7. Trusted results are visible in scanning report

TCP port	Service	SYN result	NULL result	FIN result	Xmas result	ACK result	Window result
22	ssh	open					open
111	rpcbind	open					open
514	shell	filtered					open

5.1.2. Default scan and tests

The default scan method scans 1,000 TCP ports, 1,000 UDP ports, 256 IP protocols and also implements operation system and version detection, and a full report of this scanning is shown in Appendix 2. With this, scanning took 13 minutes and 45 seconds.

The result of default scan is wider and provides more information on the host. In Table 8, it is shown that default scan also utilizes version scanning to get more information about the open ports. Also, results from UDP scanning are taken into the same table. It is worth noting that the default port for HTTP service is 80 and originally port 902 would be used for iss-realsecure service according to Nmap's database. Here it is discovered that these services do not run on default ports. There is an HTTP service found behind ports 8899 and 50000, and port 902 is utilized by VMware authentication Daemon. Open SSH port was already discovered with quick scan, but here it is discovered that SSH service's version is OpenSSH 6.6.1 and it uses SSH protocol version 2.0. This is vital information at the beginning of security testing.

Table 8. Port scanning results with default scan

Protocol	Port number	State	Service	Version
TCP	22	open	ssh	OpenSSH 6.6.1 (protocol 2.0)
TCP	111	open	rpcbind	2-4 (RPC #100000)
TCP	514	filtered	shell	
TCP	902	open	vmware-auth	VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
TCP	8899	open	http	Tornado httpd 4.3
TCP	50000	open	http	Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)
UDP	111	open	rpcbind	2-4 (RPC #100000)
UDP	500	open	isakmp	

When the service behind open port is determined, it is possible to get enhanced information to plan security testing. One way to get more information is to search for possible vulnerabilities from external databases. In addition to test tools, automated scanning and testing implementation suggests the user check the common vulnerabilities and exposures dictionary and provides a straight search link as shown in

Table 9. The link leads the user to search results with the service's name and version.

This scanning method also implements security tests with the Nmap scripting engine and Nikto. This functionality's run time is highly dependent on found services and implemented tests. In this case, services SSH, rpcbind, vmware-auth and HTTP in two ports were tested with NSE along with discovery module. The two HTTP services were also tested with Nikto. These tests took one hour, four minutes and 44 seconds.

Table 9. Suggested testing tools and external links

Type	**Port number**	**Service**	**Suggested test tools**	**External links**
TCP	22	ssh	Hydra	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= OpenSSH+6.6.1
TCP	902	vmware-auth		CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= VMware+Authentication+Daemon+1.10
TCP	8899	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Tornado+httpd+4.3
TCP	50000	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Apache+httpd+2.4.6

Analysis of the test has to be performed manually, because various pieces of information have to be analysed individually and then put together. For example, it was discovered in the scanning results that SUT uses VMware authentication service. VMware [38] is a virtualization tool to create virtual desktop environments to a physical machine. It is possible that the target uses virtualization with port forwarding to these virtual machines. Related findings from NSE scripts are shown in Table 10. First there is qscan script from the discovery module. The script sends packets to open and closed ports to calculate the average of round trip times (RTT). With these calculations, the script is able to put ports to different groups, "families", according to RTTs. Here, there are three families: number zero contains three open ports, number one contains one closed port and number two contains two open ports with longer average of RTT. Another script that is shown in Table 10 is http-traceroute which is run to ports 8899 and 50000. The script http-traceroute checks for port rules and tries to detect the presence of proxies. The script has found that both ports – 8899 and 50000 – might have a reverse proxy. Now there are three pieces of information as shown in Figure 19: VMware authentication service is in use, ports 8899 and 50000 are assigned to a different family from other ports and there is a possible reverse proxy behind these port. With this information, it can be ascertained that these two ports are behind port forwarding and running on VMware virtual machine.

5.1.3. Comprehensive scan and tests

The widest scanning method for this implementation, comprehensive scan, scans all 65,535 TCP ports, 1,000 UDP ports, 256 IP protocols and also implements operation system and version detection. The full report of this scanning is shown in Appendix 3. In SECURED case, the scanning took 12 hours, five minutes and 22 seconds. Also, comprehensive scanning implements security tests, and the security tests took one hour, 33 minutes and 29 seconds to run. Again, this time variable depends greatly on the test target.

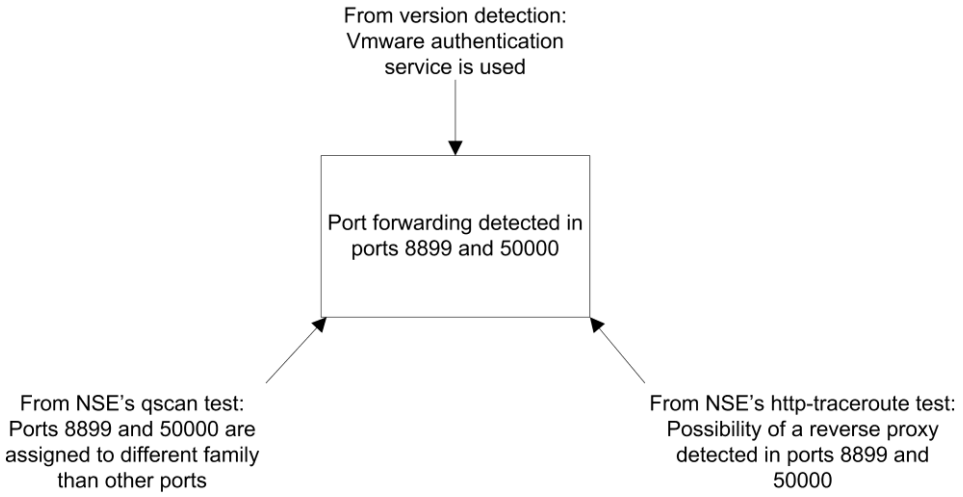

Figure 19. Three pieces of information is used to detect port forwarding in SUT

Table 10. NSE results relating to port forwarding

<table>
<tr><th>Target port</th><th>Script name</th><th>Script output</th></tr>
<tr><td>SUT in general</td><td>qscan</td><td>
<table>
<tr><th>PORT</th><th>FAMILY</th><th>MEAN (μs)</th><th>STDDEV</th><th>LOSS(%)</th></tr>
<tr><td>22</td><td>0</td><td>812.60</td><td>53.21</td><td>0.0%</td></tr>
<tr><td>111</td><td>0</td><td>903.60</td><td>348.95</td><td>0.0%</td></tr>
<tr><td>235</td><td>1</td><td>1000758.70</td><td>4153.79</td><td>0.0%</td></tr>
<tr><td>902</td><td>0</td><td>759.20</td><td>102.40</td><td>0.0%</td></tr>
<tr><td>8899</td><td>2</td><td>1305.20</td><td>201.65</td><td>0.0%</td></tr>
<tr><td>50000</td><td>2</td><td>1318.60</td><td>340.51</td><td>0.0%</td></tr>
</table>
</td></tr>
<tr><td>8899</td><td>http-traceroute</td><td>Possible reverse proxy detected.</td></tr>
<tr><td>50000</td><td>http-traceroute</td><td>Possible reverse proxy detected.</td></tr>
</table>

Table 11. Port scanning results with comprehensive scan

Protocol	Port number	State	Service	Version
TCP	22	open	ssh	OpenSSH 6.6.1 (protocol 2.0)
TCP	111	open	rpcbind	2-4 (RPC #100000)
TCP	514	filtered	shell	
TCP	902	open	vmware-auth	VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
TCP	8899	open	http	Tornado httpd 4.3
TCP	50000	open	http	Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)
TCP	60000	open	http	WSGIServer 0.1 (Python 2.7.6)
UDP	111	open	rpcbind	2-4 (RPC #100000)
UDP	500	open	isakmp	

Comprehensive scan is the widest scanning and testing method in this implementation. Therefore after the scan, it is not rational to do further scanning as all possible information is derived from the SUT. In the case of NED, there is one additional open port discovered with comprehensive scan and the results are shown in Table 11. The port is number 60000 and there is a web server running behind that port. This open port provides one more attack interface to give an opportunity to compromise the system.

5.1.4. Comparison of scanning and testing results

There are three important aspects when doing useful and helpful scanning and security testing: time scheduling, coverage of scanning and testing, and reporting. The scheduling should be planned in such a way that the penetration tester can get results with a short waiting time or during lunch, other meetings or tasks. Also, it is crucial to get comprehensive results, so there is all available information about the ports gathered from the SUT. This comprehensive scanning can be run outside working hours. Results from scanning and security tests should be reported in such a way that the penetration tester is able to start the actual testing process easily and there are already plans for testing.

Use of time with different scanning methods is quite good when compared to the timing categories in Chapter 4. A comparison of the time constraint with different scanning methods is shown in Table 12. Quick scan was run in less than two minutes so it is possible to get the first scanning results with a short waiting time. Default scan with security tests took one hour, 18 minutes and 29 seconds. The time is a little too long to get results during a lunch break. Still, the time is adequate for a situation where the penetration tester focuses on do some other tasks during scanning and can read the results after two or three hours. Comprehensive scanning and security testing is intended to be done outside working hours. With eight hours daily working time, the time between leaving the work place one afternoon and continuing working the next morning is 16 hours. That can be considered the maximum time for this scanning method. In this case, the full scanning and testing time is 13 hours, 38 minutes and 51 seconds, which it is possible to run during evening and night.

Table 12. Scanning and testing time by scanning method

	Quick	Default	Comprehensive
Port scanning	1m 11s	13m 45s	12h 5min 22s
Security tests	Not run	1h 4m 44s	1h 33min 29s
Total	1m 11s	1h 18m 29s	13h 38m 51s

A summary of coverage of scanning methods is shown in Table 13. It is shown that there are more findings when the scanning method is changed from quick to default and from default to comprehensive. When a default scan is used instead of a quick scan, the number of trusted TCP results is doubled and also two UDP ports are discovered by the default scan. When a comprehensive scan was executed, one additional TCP port was discovered. It is worth noting that there are six open or filtered TCP ports among the 1,000 most commonly used ports and only one among the remaining 64,535 ports. It is still crucial to scan all TCP ports, because the penetration tester has to find all attack interfaces, and make sure that there are no security flaws or vulnerabilities in the SUT.

Table 13. Summary of results with different scanning and testing methods

	Quick	Default	Comprehensive
Scanned TCP ports	100	1000	65535
Trusted TCP results	3	6	7
Scanned UDP ports	Not run	1000	1000
Trusted UDP results	Not run	2	2
Services with HTTP tests	Not run	2	3
Services with NSE	Not run	5	6

The categorization of results, which is described in Chapter 4.4, give a clear and detailed presentation of the scan and test results. General info and basic scanning results give a good understanding of the target system as network addresses, port states, used services and suggested testing tools are shown in one or two A4 sized pages. Additional scanning results can be used when deeper analysis is required to create testing scenarios.

The report from security tests is longer and more detailed than the scanning report. Default scan run HTTP tests on two services and NSE on five services. With this testing coverage, the report is about seven pages long. This requires time from the penetration tester to analyse. Still, the results are shown in clear tables, so results can be reviewed easily in the same document.

5.2. Other test targets

The scanning and security testing implementation was also tested against two other test targets. The approach for this testing is fully black-box: there is a minimal amount of information about the target hosts. Only IP addresses to the targets and network configuration to access them are known beforehand. In some cases, this might be the situation when penetration testing is started. The preliminary information on the test environment is shown in Figure 20.

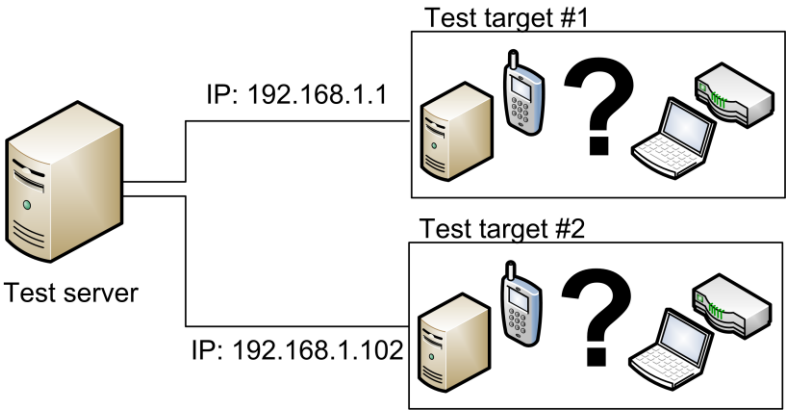

Figure 20. Preliminary information of testing unknown targets

A summary of the scanning results is shown in Table 14 for both test targets. The first target seems to be a Residential Gateway Device. The scanning and testing implementation revealed that the device is manufactured by Linksys Inc. and the model of the device is WRT54GL version 4.30.7. By searching the internet, it is possible to find the manual for the device, which reveals that the device is a wireless broadband router. There are two TCP ports, three UDP ports and three IP protocols open or filtered in the router. The operating system of the device is Linux 2.4.18 – 2.4.35, and it is likely embedded regarding the results. This gives quite a lot of information on the host, although there was minimal information before scanning and testing. There are five possible vulnerabilities found from the target. These vulnerabilities are related to authentication and HTTP header in the target's HTTP service, but these vulnerabilities should be confirmed manually. Also, criticality of the vulnerability should be determined.

The second target is not as easy to identify as the first. In the scanning module, it is determined that the target has the Windows operating system. The version of Windows can be 2003 or XP. If Windows XP is used in the target system, Service Pack (SP) can be SP2 or SP3. Port scanning reveals more information about the operating system: port 135 with Microsoft Windows RPC, port 139 with Microsoft Windows 98 netbios-ssn and port 445 with Microsoft Windows XP microsoft-ds are open. With this information, it can be determined that Microsoft Windows is definitely used in the test target and the version is likely to be XP. The vendor of the network card is Dell so it can be ascertained that the target system is a laptop or desktop computer that uses the Microsoft Windows XP operating system and is manufactured by Dell.

There were five open TCP ports and one used IP protocol discovered from test target #2. Three possible vulnerabilities were found, which are related to the HTTP server running behind port 2869.

Table 14. Summary of test results with two unknown targets

	Target #1	**Target #2**
General info:		
IP address	192.168.1.1	192.168.1.102
Scanning time with quick scan	1 minute 38 seconds	2 minutes 19 seconds
...default scan and testing	1 hour 13 minutes 44 seconds	1 hour 2 minutes 8 seconds
...comprehensive scan and testing	15 hours 5 minutes and 27 seconds	13 hours 41 minutes 13 seconds
Scanning module:		
IPv4 address	192.168.1.1	192.168.1.102
MAC address	68:7F:74:0A:6C:8E	00:0D:56:7D:EF:45
Vendor	Cisco-Linksys	Dell
Operating system	Linux 2.4.18 - 2.4.35 (likely embedded)	Microsoft Windows; server 2003 SP0, XP SP2, XP SP3 or Small Business Server 2003
Non-closed TCP ports	2 ports (ports 80 and 5431)	5 ports (ports 135, 139, 445, 2869 and 6502)
Non-closed UDP ports	3 ports (ports 67, 69, 1900)	0 ports
Used IP protocols	3 protocols (ICMP, TCP, UDP)	1 protocol (ICMP)
Testing module:		
Executed security tests	NSE: HTTP scripts and discovery module Nikto: HTTP service behind port 80	NSE: HTTP scripts and discovery module Nikto: HTTP service behind port 2869
Possible vulnerabilities	Five	Three
Other information retrieved	Name: Residential Gateway Device Manufacturer: Linksys Inc. Model Descr: Internet Access Server Model Name: WRT54GL Model Version: v4.30.7	OS: Windows XP (Windows 2000 LAN Manager) OS Common Platform Enumeration (CPE): cpe:/o:microsoft:windows_xp::

5.3. Component in Network Edge Device (NED)

SECURED system is designed to include all security protection in one device: Network Edge Device. The heart of NED' architecture (shown in Figure 21) is Trusted Virtual Domain (TVD), which is in charge of the user's network traffic and security applications. TVD includes Personal Security Application (PSA) and Personal Security Controller (PSC) for every user. PSAs contain security control processes like "block mature content" or "check for malware". It is possible that one

user has multiple PSAs as one PSA is responsible for one security controller. The PSAs' behaviour is monitored and controlled by PSC. [39]

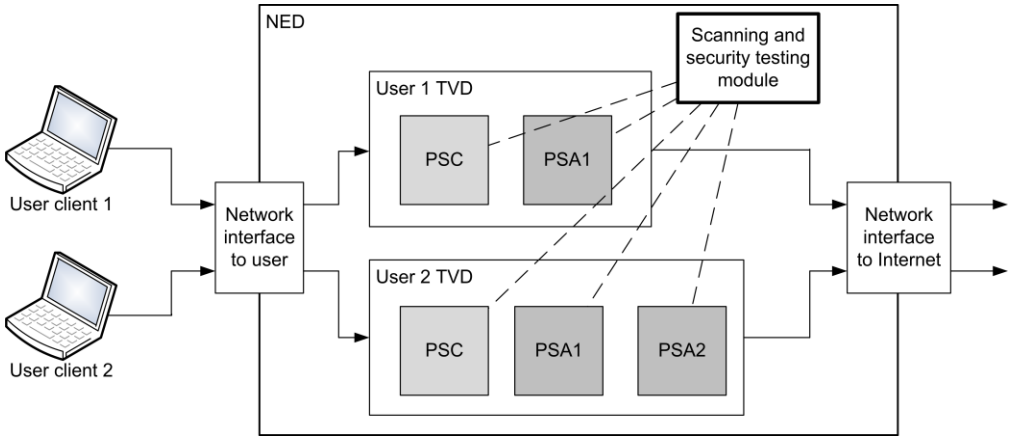

Figure 21. NED's architecture, the user's network connections are shown with a solid line and scanning and security testing module's connections with a dashed line

The PSAs and PSCs contain sensitive user data and all users' traffic is flown through PSAs. Although the NED is secured from outside attacks, PSAs and PSCs should be secured from inside the NED to make sure that no user data is leaked to a malicious user. Automated scanning and testing implementation that is developed in this thesis can be used to verify PSAs' and PSCs' security as shown in Figure 21. The scanning and testing has to be done once in a while, because configuration of PSAs and PSCs change when different users log in and out. The scanning and testing implementation needs to dynamically search IP addresses and network interfaces for each PSA and PSC, because the number of these components and their addresses are changed according to number of users.

The motivation of scanning the NED's inner components lies in PSA security. It is possible that PSAs are developed not only in SECURED project but by a third party developer. In that case, it is important to verify that the PSAs are secure and there are no intentional spying or misuse scenarios with the PSA. The scanning for open interfaces and running security tests can be used to ensure PSA's safety and integrity. The scanning and security testing is also possible when the PSA has been taken into use.

By deploying the automated scanning and security testing implementation as a component in NED, the fact that it is possible to deploy the implementation in various scenarios is illustrated. Within SECURED project, it is possible to do scanning and testing both outside and inside NED. Although the implementation is originally designed to test a single network host, it is possible to run the scripts to many targets. As in security testing in general, also in this case it is crucial for the tester to have permission to install this implementation to the NED and run scan and tests towards PSAs and PSCs. The test results should only be used to develop NED, PSAs and PSCs to be more secure, and publishing the results should be considered only when it does not harm the NED's development.

6. SUMMARY

The scope of this thesis was to develop software that helps black-box penetration testing by automating port scanning and security tests of TCP and UDP ports and IP protocols. The goal was to develop a comprehensive port scanning method to achieve inclusive results when there is no preliminary information on the system under test. Also, quicker scanning and testing methods were developed, so it is possible to get results in a shorter time, for example during a lunch break or a meeting. After port scanning is executed, various security tests are run towards found services on the target host. The implementation also provides test planning by suggesting appropriate test tools and giving links to external databases. The implementation was tested in SECURED project with Network Edge Device and, finally this automated scanning and security testing implementation was applied to become a component in the NED.

Port scanning was performed with Nmap. There were six different TCP port scanning methods used to interrogate the target system with different TCP packets. When many scanning methods are used, it is crucial to determine which results can be used so as to determine final port state and which are not. In this thesis, the scanning methods' port assignments were divided to trusted and untrusted assignments.

There were three scheduling options to perform port scanning: quick, default and comprehensive. With quick scanning, it is possible to get results from SUT in a short waiting time so the tester can start planning further tests right away. Default scan is intended to keep running during a lunch break, meetings or tester can do other work tasks during scanning and testing. Comprehensive scan is really time consuming and should be executed outside working hours.

Test module was implemented along with port scanning methods to start security tests after scanning was performed. Test module uses test library to execute tests towards services that were found from the SUT in port scanning. There were two scripts implemented to test library: Nmap scripting engine and HTTP tests. The Nmap scripting engine runs scripts both in general to get more information from the target system and towards services that were discovered in port scanning. HTTP tests implement the Nikto web application and server scanning tool. HTTP tests are run towards ports which have a HTTP service behind them.

This implementation had four main focuses: time usage in different scanning methods, getting trusted results when different scanning methods are used, scanning coverage in different methods and reporting of scanning and testing results. Time usage was under two minutes with quick scan, less than one and a half hours with default scan and over 13 hours with comprehensive scan. These timings were good to be performed in different situations during the tester's working day. In TCP port scanning, the port state assignments from six different scanning methods were researched and divided to trusted and untrusted assignments. The trusted assignments were used to determine the final port state. It was discovered that using of different scanning methods was useful to get more comprehensive scanning results. Although six of seven open or filtered TCP ports, which were found from Network Edge Device, were among the 1,000 most commonly used ports, it is important to scan all ports to discover all possible attack interfaces. The results from scanning and security testing were assembled into a clear single report along with suggestions and guidance to create a test plan.

The automated scanning and security testing implementation was also implemented to Network Edge Device under SECURED project. The function of the implementation is to scan and test the user's Personal Security Applications and Personal Security Controllers.

7. REFERENCES

[1] Cai J., Zou P., Xiong D., and He J. (2015) A guided fuzzing approach for security testing of network protocol software. 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS) pp. 726–729, DOI: 10.1109/ICSESS.2015.7339160.

[2] Shahriar H. and Zulkernine M. (2009) Automatic Testing of Program Security Vulnerabilities. 2009 33rd Annual IEEE International Computer Software and Applications Conference vol. 2 pp. 550–555, DOI: 10.1109/COMPSAC.2009.191.

[3] Brumley D., Newsome J., and Song D. (2006) Towards automatic generation of vulnerability-based signatures. 2006 IEEE Symposium on Security and Privacy (S&P'06) p. 15 pp.–16, DOI: 10.1109/SP.2006.41.

[4] Scarfone K. and Souppaya M. (2008) Technical guide to information security testing and assessment. NIST Special Publication.

[5] Engebretson P. (2013) Basics of Hacking and Penetration Testing. Elsevier Science.

[6] Lyon G. F. (2008) Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Com LLC.

[7] SECURED project (2016) The SECURED project (SECURity at the network EDge) — SECURED (SECURity at the network EDge). [Online]. Available: http://www.secured-fp7.eu/. [Accessed: 21-Mar-2016].

[8] Potter B. and McGraw G. (2004) Software security testing. IEEE Security & Privacy Magazine vol. 2, no. 5 pp. 81–85, DOI: 10.1109/MSP.2004.84.

[9] Muller A., Meucci M., Keary E., and Cuthbert D. (2015) OWASP Testing Guide 4.0.

[10] Weidman G. (2014) Penetration Testing: A Hands-On Introduction to Hacking. No Starch Press.

[11] Xue Qiu, Shuguang Wang, Qiong Jia, Chunhe Xia, and Qingxin Xia (2014) An automated method of penetration testing. 2014 IEEE Computers, Communications and IT Applications Conference pp. 211–216, DOI: 10.1109/ComComAp.2014.7017198.

[12] Prandini M. and Ramilli M. (2010) Towards a practical and effective security testing methodology. The IEEE symposium on Computers and Communications pp. 320–325, DOI: 10.1109/ISCC.2010.5546813.

[13] The MITRE Corporation (2015) CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC). [Online]. Available: https://capec.mitre.org/. [Accessed: 18-Nov-2015].

[14] The MITRE Corporation (2013) CVE Introductory Brochure. [Online]. Available: http://makingsecuritymeasurable.mitre.org/docs/cve-intro-handout.pdf. [Accessed: 30-Dec-2015].

[15] The MITRE Corporation (2013) CWE Introductory Brochure. [Online].

Available: http://makingsecuritymeasurable.mitre.org/docs/cwe-intro-handout.pdf. [Accessed: 30-Dec-2015].

[16] The MITRE Corporation (2015) Common weakness enumeration list version 2.9. [Online]. Available: https://cwe.mitre.org/data/index.html. [Accessed: 30-Dec-2015].

[17] Provos N. (2011) Firewall. in Encyclopedia of Cryptography and Security, H. van Tilbork and S. Jajodia, Eds. Springer US, pp. 471–474, DOI: 10.1007/978-1-4419-5906-5.

[18] Maurushat A. (2013) Criminal Offences: Unauthorised Access, Modification or Interference Comprovisions. in Disclosure of Security Vulnerabilities, Springer London, pp. 35–53.

[19] Panjwani S., Tan S., Jarrin K. M., and Cukier M. (2005) An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack. 2005 International Conference on Dependable Systems and Networks (DSN'05) pp. 602–611, DOI: 10.1109/DSN.2005.18.

[20] Ghanem W. A. H. M. and Belaton B. (2013) Improving accuracy of applications fingerprinting on local networks using NMAP-AMAP-ETTERCAP as a hybrid framework. 2013 IEEE International Conference on Control System, Computing and Engineering pp. 403–407, DOI: 10.1109/ICCSCE.2013.6719998.

[21] Cotton M., Eggert L., Touch J., Westerlund M., and Cheshire S. (2011) Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry.

[22] Gu Y., Fu Y., Prakash A., Lin Z., and Yin H. (2014) Multi-Aspect, Robust, and Memory Exclusive Guest OS Fingerprinting. IEEE Transactions on Cloud Computing vol. 2, no. 4 pp. 380–394, DOI: 10.1109/TCC.2014.2338305.

[23] Allen J. (2007) OS and Application Fingerprinting Techniques. SANS Institute InfoSec Reading Room.

[24] Kyounghee K., Pilyong K., and Wontae S. (2007) Computational Science and Its Applications – ICCSA 2007. vol. 4706. Berlin, Heidelberg: Springer Berlin Heidelberg, DOI: 10.1007/978-3-540-74477-1.

[25] Gagnon F., Esfandiari B., and Bertossi L. (2007) A Hybrid Approach to Operating System Discovery using Answer Set Programming. 2007 10th IFIP/IEEE International Symposium on Integrated Network Management pp. 391–400, DOI: 10.1109/INM.2007.374804.

[26] Medeiros J. P. S., da Cunha A. C., Brito A. M., and Motta Pires P. S. (2007) Automating security tests for industrial automation devices using neural networks. 2007 IEEE Conference on Emerging Technologies & Factory Automation (EFTA 2007) pp. 772–775, DOI: 10.1109/EFTA.2007.4416854.

[27] Mathew K., Tabassum M., and Lu Ai Siok M. V. (2014) A study of open ports as security vulnerabilities in common user computers. 2014 International Conference on Computational Science and Technology (ICCST) pp. 1–6,

DOI: 10.1109/ICCST.2014.7045193.

[28] Lyon G. (2011) The Official Nmap Project Guide to Network Discovery and Security Scanning. [Online]. Available: http://www.nmap.org/book/toc.html. [Accessed: 18-Sep-2015].

[29] Postel J. (1981) RFC 793 Transmission Control Protocol. IETF, DOI: 10.17487/RFC0793.

[30] Zhang X., Knockel J., and Crandall J. R. (2015) Original SYN: Finding machines hidden behind firewalls. 2015 IEEE Conference on Computer Communications (INFOCOM) pp. 720–728, DOI: 10.1109/INFOCOM.2015.7218441.

[31] Postel J. (1981) RFC 792 Internet control message protocol. IETF, DOI: 10.17487/RFC0792.

[32] Quina A. and Stavliotis L. (2015) SPARTA - Network Infrastructure Penetration Testing Tool. [Online]. Available: http://sparta.secforce.com/. [Accessed: 29-Sep-2015].

[33] Li L., Dong Q., Liu D., and Zhu L. (2013) The Application of Fuzzing in Web Software Security Vulnerabilities Test. 2013 International Conference on Information Technology and Applications pp. 130–133, DOI: 10.1109/ITA.2013.36.

[34] CIRT.net (2016) Nikto2 | CIRT.net. [Online]. Available: https://cirt.net/Nikto2. [Accessed: 26-Feb-2016].

[35] Dalton C., Lioy A., Lopez D., and Risso F. (2014) Exploiting the network for securing personal devices. vol. 470. Cham: Springer International Publishing, DOI: 10.1007/978-3-319-12574-9.

[36] Lioy A., Pastor A., Risso F., Sassu R., and Shaw A. L. (2014) Offloading Security Applications into the Network. *eChallenges e-2014, 2014 Conference*. pp. 1–9.

[37] Hauser V. and Kessler R. (2014) THC-Hydra | Penetration Testing Tools. [Online]. Available: http://tools.kali.org/password-attacks/hydra. [Accessed: 08-Mar-2016].

[38] VMware Inc. (2016) VMware Virtualization for Desktop & Server, Application, Public & Hybrid Clouds. [Online]. Available: http://www.vmware.com/. [Accessed: 09-Mar-2016].

[39] Bonafiglia R., Ciaccia F., Lioy A., Nemirovsky M., Risso F., and Su T. (2015) Offloading personal security applications to a secure and trusted network node. Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) pp. 1–2, DOI: 10.1109/NETSOFT.2015.7116171.

8. APPENDICES

Appendix 1. NED scanning results with quick scan

Appendix 2. NED scanning and security testing results with default scan

Appendix 3. NED scanning and security testing results with comprehensive scan

Appendix 1. NED scanning results with quick scan

```
Results of scanning 100 TCP ports
Executed at 07 March 2016 13:51:47

General info
------------

IPv4 address: 130.188.91.224
IPv6 address:
MAC address:
Vendor: None
Scan runtime: 0 hours 01 minutes and 11 seconds

BASIC SCANNING RESULTS
----------------------
```

Count of scanned TCP ports

Scan method	Open	Closed	Filtered	Unfiltered	Open|filtered
Syn	2	97	1	0	0
Null	0	0	0	0	100
Fin	0	0	0	0	100
Xmas	0	0	0	0	100
Ack	0	0	0	100	0
Window	100	0	0	0	0

Found ports

TCP port	Service	Deducted state	Scanning method	Suggested testing tools
22	ssh	open	syn	Hydra
111	rpcbind	open	syn	
514	shell	filtered	syn	

```
ADDITIONAL SCANNING RESULTS
---------------------
```

TCP port	Service	SYN result	NULL result	FIN result	Xmas result	ACK result	Window result
22	ssh	open					open
111	rpcbind	open					open
514	shell	filtered					open

Appendix 2. NED scanning and security testing results with default scan

Results of scanning 1000 TCP ports, 1000 UDP ports and IP protocols
Executed at 07 March 2016 13:53:03

General info

IPv4 address: 130.188.91.224
IPv6 address:
MAC address:
Vendor: None
Scan runtime: 0 hours 13 minutes and 45 seconds

OS scanning

Operating system	Accuracy
Microsoft Windows 7 or Windows Server 2012	100
Microsoft Windows XP SP3	100

BASIC RESULTS

Count of scanned TCP ports

Scan method	Open	Closed	Filtered	Unfiltered	Open|filtered
Syn	5	994	1	0	0
Null	0	0	0	0	1000
Fin	0	0	0	0	1000
Xmas	0	0	0	0	1000
Ack	0	0	0	1000	0
Window	1000	0	0	0	0
UDP	2	0	0	0	998
IP protocol	2	0	252	0	2

Found ports

Protocol	Port number	State	Service	Version
TCP	22	open	ssh	OpenSSH 6.6.1 (protocol 2.0)
TCP	111	open	rpcbind	2-4 (RPC #100000)
TCP	514	filtered	shell	
TCP	902	open	vmware-auth	VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
TCP	8899	open	http	Tornado httpd 4.3
TCP	50000	open	http	Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)
UDP	111	open	rpcbind	2-4 (RPC #100000)
UDP	500	open	isakmp	

IP protocols

IP protocol	Service	Protocol status
1	icmp	open
6	tcp	open
17	udp	open|filtered
47	gre	open|filtered

Suggested testing tools

Type	Port number	Service	Suggested test tools	External links
TCP	22	ssh	Hydra	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= OpenSSH+6.6.1
TCP	902	vmware-auth		CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= VMware+Authentication+Daemon+1.10
TCP	8899	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Tornado+httpd+4.3
TCP	50000	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Apache+httpd+2.4.6

ADDITIONAL RESULTS

TCP port scanning results with all scanning methods

TCP port	Service	SYN result	NULL result	FIN result	Xmas result	ACK result	Window result
22	ssh	open					open
111	rpcbind	open					open
514	shell	filtered					open
902	iss-realsecure	open					open
8899	ospf-lite	open					open
50000	snet-sensor-mgmt	open					open

TCP open and filtered ports' deduction table

TCP port	Service	Deducted state	Scanning method
22	ssh	open	syn
111	rpcbind	open	syn
514	shell	filtered	syn
902	iss-realsecure	open	syn
8899	ospf-lite	open	syn
50000	ibm-db2	open	syn

Responses from version scanning with not identified version

Protocol	Port number	Service	Return data
UDP	500	isakmp	SF-Port500-UDP:V=7.01%I=7%D=3/2%Time=56D6DBD7%P=x86_64-pc-linux-gnu%r(RPCCheck,24,"r\xfe\x1d\x13\0\0\0\0\0\0\0\x02\0\x01\x86\xa0\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(DNSVersionBindReq,24,"\0\x06\x01\0\0\x01\0\0\0\0\0\0\x07ver\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NBTStat,24,"\x80\xf0\0\x10\0\x01\0\0\0\0\0\0\x20CKA\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(SIPOptions,24,"OPTIONS\x20sip:nm\x20S\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NTPRequest,24,"\xe3\0\x04\xfa\0\x01\0\0\0\x01\0\0\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\

			$\0\0\0\x08\0\0\0\x05")%r(SNMPv1public,24,"0\x82\0/\x02\x01\0\x04\x06public\xa0\)\x20%\(\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(AFSVersionRequest,24,"\0\0\x03\xe7\0\0\0\0\0\0\0e\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(DNS-SD,24,"\0\0\0\0\0\x01\0\0\0\0\0\0\t_se\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(Citrix,24,"\x1e\0\x010\x02\xfd\xa8\xe3\0\0\0\0\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(Kerberos,24,"j\x81n0\x81k\xa1\x03\x02\x01\x05\xa2\x03\x02\x01\n\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(sybaseanywhere,24,"\x1b\0\0=\0\0\0\0\x12CONNECT\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NetMotionMobility,24,"\0@P\0\0\0\0\x85\]\xb4\x91\(\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05"

Results of automated tests

Executed at 07 March 2016 14:06:49

Host info

IPv4 address: 130.188.91.224
IPv6 address:
MAC address:
Vendor: None
Scan runtime: 1 hours 04 minutes and 44 seconds

Nikto test results

Nikto test results on host 130.188.91.224:8899

1 The anti-clickjacking X-Frame-Options header is not present. 2 The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS 3 The site uses SSL and the Strict-Transport-Security HTTP header is not defined. 4 The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

Nikto test results for host 130.188.91.224:50000

1 Connection failed to the web server

Nmap scripting engine test results for the host

NSE discovery category scripts

ScriptID	Script Output
broadcast-	IP: 192.168.182.2 MAC: 00:50:56:f2:e4:85

ping	Use --script-args=newtargets to add the results as targets
targets-asn	targets-asn.asn is a mandatory parameter
targets-ipv6-multicast-slaac	IP: fe80::5d32:27e5:6b3c:fb97 MAC: 00:50:56:c0:00:08 IFACE: eth0 IP: fe80::b479:9c0c:faa4:96e9 MAC: 00:50:56:c0:00:08 IFACE: eth0 Use --script-args=newtargets to add the results as targets
asn-query	BGP: 130.188.0.0/16 | Country: FI Origin AS: 565 - Technical Research Centre of Finland,FI Peer AS: 1741
dns-brute	Can't guess domain of "verifier"; use dns-brute.domain script argument.
fcrdns	PASS (224.91.vtt.fi)
hostmap-ip2hosts	hosts: Error: could not GET http://www.ip2hosts.com/csv.php?ip=130.188.91.224
hostmap-robtex	
ip-geolocation-geobytes	ERROR: Script execution failed (use -d to debug)
ip-geolocation-geoplugin	ERROR: Script execution failed (use -d to debug)
ip-geolocation-maxmind	ERROR: Script execution failed (use -d to debug)
ipidseq	Unknown
path-mtu	PMTU == 1500
qscan	PORT FAMILY MEAN (us) STDDEV LOSS (%) 22 0 812.60 53.21 0.0% 111 0 903.60 348.95 0.0% 235 1 1000758.70 4153.79 0.0% 902 0 759.20 102.40 0.0% 8899 2 1305.20 201.65 0.0% 50000 2 1318.60 340.51 0.0%
whois-domain	You should provide a domain name.
whois-ip	Record found at whois.ripe.net inetnum: 130.188.0.0 - 130.188.255.255 netname: VTTNET descr: Technical Research Centre of Finland country: FI orgname: VTT Technical Research Centre of Finland organisation: ORG-VTRC1-RIPE email: info@vtt.fi

Nmap scripting engine test results for TCP ports

NSE results for TCP service ssh (OpenSSH 6.6.1 (protocol 2.0)) in port 22

ScriptID	Script output
banner	SSH-2.0-OpenSSH_6.6.1
ssh-hostkey	2048 c7:e1:77:b3:91:b1:88:9d:12:4d:18:e9:ec:0c:c6:b5 (RSA) 256 77:07:f5:48:1f:0a:94:3f:2f:52:95:f8:b1:de:41:8d

	(ECDSA)
ssh2-enum-algos	kex_algorithms: (8) curve25519-sha256@libssh.org ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 diffie-hellman-group-exchange-sha256 diffie-hellman-group-exchange-sha1 diffie-hellman-group14-sha1 diffie-hellman-group1-sha1 server_host_key_algorithms: (3) ssh-rsa ecdsa-sha2-nistp256 ssh-ed25519 encryption_algorithms: (16) aes128-ctr aes192-ctr aes256-ctr arcfour256 arcfour128 aes128-gcm@openssh.com aes256-gcm@openssh.com chacha20-poly1305@openssh.com aes128-cbc 3des-cbc blowfish-cbc cast128-cbc aes192-cbc aes256-cbc arcfour rijndael-cbc@lysator.liu.se mac_algorithms: (19) hmac-md5-etm@openssh.com hmac-sha1-etm@openssh.com umac-64-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512-etm@openssh.com hmac-ripemd160-etm@openssh.com hmac-sha1-96-etm@openssh.com hmac-md5-96-etm@openssh.com hmac-md5 hmac-sha1 umac-64@openssh.com umac-128@openssh.com hmac-sha2-256 hmac-sha2-512 hmac-ripemd160 hmac-ripemd160@openssh.com hmac-sha1-96 hmac-md5-96 compression_algorithms: (2) none zlib@openssh.com

NSE results for TCP service rpcbind (2-4 (RPC #100000)) in port 111

ScriptID	Script output
rpcinfo	program version port/proto service 100000 2,3,4 111/tcp rpcbind 100000 2,3,4 111/udp rpcbind

NSE results for TCP service vmware-auth (VMware Authentication Daemon 1.10 (Uses VNC, SOAP)) in port 902

ScriptID	Script output
banner	220 VMware Authentication Daemon Version 1.10: SSL Required,...

NSE results for TCP service http (Tornado httpd 4.3) in port 8899

ScriptID	Script output
http-brute	Path "/" does not require authentication
http-chrono	Request times for /; avg: 163.44ms; min: 160.30ms; max: 168.97ms
http-comments-displayer	Couldn't find any comments.
http-cross-domain-policy	ERROR: Script execution failed (use -d to debug)
http-csrf	Couldn't find any CSRF vulnerabilities.
http-date	Mon, 07 Mar 2016 12:07:49 GMT; +13s from local time.
http-devframework	ERROR: Script execution failed (use -d to debug)
http-dombased-xss	Couldn't find any DOM based XSS.
http-errors	Couldn't find any error pages.
http-feed	Couldn't find any feeds.
http-fetch	Please enter the complete path of the directory to save data in.
http-fileupload-exploiter	
http-frontpage-login	false
http-headers	Date: Mon, 07 Mar 2016 13:09:17 GMT Content-Length: 36 Etag: "26cb994701b88377ba04469c3d76d8cc2aefbe22" Content-Type: application/json Server: TornadoServer/4.3 (Request type: GET)
http-malware-host	Host appears to be clean
http-methods	Supported Methods: GET
http-mobileversion-checker	No mobile version detected.
http-referer-checker	Couldn't find any cross-domain scripts.
http-server-header	TornadoServer/4.3

http-sitemap-generator	Directory structure:
	  /
	  Other: 1
	Longest directory structure:
	  Depth: 0
	  Dir: /
	Total files found (by extension):
	  Other: 1

http-slowloris	false
http-stored-xss	Couldn't find any stored XSS vulnerabilities.

http-title	Site doesn't have a title (application/json).
http-traceroute	Possible reverse proxy detected.
http-useragent-tester	Allowed User Agents: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) libwww lwp-trivial libcurl-agent/1.0 PHP/ Python-urllib/2.5 GT::WWW Snoopy MFC_Tear_Sample HTTP::Lite PHPCrawl URI::Fetch Zend_Http_Client http client PECL::HTTP Wget/1.13.4 (linux-gnu) WWW-Mechanize/1.34
http-vhosts	127 names had status 405
http-xssed	ERROR: Script execution failed (use -d to debug)
ssl-cert	Subject: commonName=verifier/organizationName=POLITO/stateOrProvinceName=Piemonte/countryName=IT Not valid before: 2015-10-28T15:21:41 Not valid after: 2016-10-27T15:21:41
ssl-date	2016-03-07T13:09:08+00:00; +12s from scanner time.
ssl-enum-ciphers	SSLv3: ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 1024) - D TLS_RSA_WITH_AES_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_AES_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_IDEA_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - A TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - A TLS_RSA_WITH_SEED_CBC_SHA (rsa 1024) - A compressors: NULL cipher preference: client warnings: CBC-mode cipher in SSLv3 (CVE-2014-3566) Ciphersuite uses MD5 for message integrity Weak certificate signature: SHA1 TLSv1.0: ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 1024) - D TLS_RSA_WITH_AES_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_AES_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_IDEA_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - A TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - A TLS_RSA_WITH_SEED_CBC_SHA (rsa 1024) - A compressors:

	NULL cipher preference: client warnings: Ciphersuite uses MD5 for message integrity Weak certificate signature: SHA1 TLSv1.1: ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 1024) - D TLS_RSA_WITH_AES_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_AES_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_IDEA_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - A TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - A TLS_RSA_WITH_SEED_CBC_SHA (rsa 1024) - A compressors: NULL cipher preference: client warnings: Ciphersuite uses MD5 for message integrity Weak certificate signature: SHA1 Weak cipher RC4 in TLSv1.1 or newer not needed for BEAST mitigation TLSv1.2: ciphers: TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 1024) - D TLS_RSA_WITH_AES_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 1024) - A TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 1024) - A TLS_RSA_WITH_AES_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 1024) - A TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_IDEA_CBC_SHA (rsa 1024) - A TLS_RSA_WITH_RC4_128_MD5 (rsa 1024) - A TLS_RSA_WITH_RC4_128_SHA (rsa 1024) - A TLS_RSA_WITH_SEED_CBC_SHA (rsa 1024) - A compressors: NULL cipher preference: client warnings: Ciphersuite uses MD5 for message integrity Weak certificate signature: SHA1 Weak cipher RC4 in TLSv1.1 or newer not needed for BEAST mitigation least strength: D
ssl-google-cert-catalog	No DB entry

NSE results for TCP service http (Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)) in port 50000

ScriptID	Script output
http-brute	Path "/" does not require authentication
http-chrono	Request times for /; avg: 156.64ms; min: 155.75ms; max: 157.71ms
http-comments-displayer	Couldn't find any comments.
http-cross-	ERROR: Script execution failed (use -d to debug)

domain-policy	
http-csrf	Couldn't find any CSRF vulnerabilities.
http-date	Mon, 07 Mar 2016 12:07:49 GMT; +13s from local time.
http-devframework	ERROR: Script execution failed (use -d to debug)
http-dombased-xss	Couldn't find any DOM based XSS.
http-errors	Spidering limited to: maxpagecount=40; withinhost=verifier Found the following error pages: Error Code: 400 http://verifier:50000/
http-feed	Couldn't find any feeds.
http-fetch	Please enter the complete path of the directory to save data in.
http-fileupload-exploiter	
http-frontpage-login	false
http-headers	Date: Mon, 07 Mar 2016 13:08:59 GMT Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips Content-Length: 362 Connection: close Content-Type: text/html; charset=iso-8859-1 (Request type: GET)
http-malware-host	Host appears to be clean
http-methods	Supported Methods: GET HEAD POST OPTIONS
http-mobileversion-checker	No mobile version detected.
http-referer-checker	Couldn't find any cross-domain scripts.
http-server-header	Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips
http-sitemap-generator	Directory structure: Longest directory structure: Depth: 0 Dir: / Total files found (by extension):
http-slowloris	false
http-stored-xss	Couldn't find any stored XSS vulnerabilities.
http-title	400 Bad Request
http-traceroute	Possible reverse proxy detected.
http-useragent-tester	Allowed User Agents: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) libwww lwp-trivial libcurl-agent/1.0 PHP/ Python-urllib/2.5 GT::WWW Snoopy MFC_Tear_Sample

	HTTP::Lite PHPCrawl URI::Fetch Zend_Http_Client http client PECL::HTTP Wget/1.13.4 (linux-gnu) WWW-Mechanize/1.34
http-vhosts	127 names had status 400
http-xssed	ERROR: Script execution failed (use -d to debug)

Appendix 3. NED scanning and security testing results with comprehensive scan

```
Results of scanning all TCP ports, 1000 UDP ports and IP protocols
Executed at 07 March 2016 15:55:38

General info
------------
IPv4 address: 130.188.91.224
IPv6 address:
MAC address:
Vendor: None
Scan runtime: 12 hours 05 minutes and 22 seconds
```

OS scanning

Operating system	Accuracy
Microsoft Windows 7 or Windows Server 2012	100
Microsoft Windows XP SP3	100

BASIC RESULTS

Count of scanned TCP ports

Scan method	Open	Closed	Filtered	Unfiltered	Open|filtered
Syn	6	65528	1	0	0
Null	0	0	0	0	65535
Fin	0	0	0	0	65535
Xmas	0	0	0	0	65535
Ack	0	0	0	65535	0
Window	65535	0	0	0	0
UDP	2	0	0	0	998
IP protocol	2	0	252	0	2

Found ports

Protocol	Port number	State	Service	Version
TCP	22	open	ssh	OpenSSH 6.6.1 (protocol 2.0)
TCP	111	open	rpcbind	2-4 (RPC #100000)
TCP	514	filtered	shell	
TCP	902	open	vmware-auth	VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
TCP	8899	open	http	Tornado httpd 4.3
TCP	50000	open	http	Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)
TCP	60000	open	http	WSGIServer 0.1 (Python 2.7.6)
UDP	111	open	rpcbind	2-4 (RPC #100000)
UDP	500	open	isakmp	

IP protocols

IP protocol	Service	Protocol status
1	icmp	open
6	tcp	open
17	udp	open|filtered
47	gre	open|filtered

Suggested testing tools

Type	Port number	Service	Suggested test tools	External links
TCP	22	ssh	Hydra	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= OpenSSH+6.6.1
TCP	902	vmware-auth		CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= VMware+Authentication+Daemon+1.10
TCP	8899	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Tornado+httpd+4.3
TCP	50000	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= Apache+httpd+2.4.6
TCP	60000	http	OWASP ZAP, Burpsuite, Nikto	CVE: http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword= WSGIServer+0.1

ADDITIONAL RESULTS

TCP port scanning results with all scanning methods

TCP port	Service	SYN result	NULL result	FIN result	Xmas result	ACK result	Window result
22	ssh	open					open
111	rpcbind	open					open
514	shell	filtered					open
902	iss-realsecure	open					open
8899	ospf-lite	open					open
50000	snet-sensor-mgmt	open					open
60000	unknown	open					open

TCP open and filtered ports' deduction table

TCP port	Service	Deducted state	Scanning method
22	ssh	open	syn
111	rpcbind	open	syn
514	shell	filtered	syn
902	iss-realsecure	open	syn
8899	ospf-lite	open	syn
50000	ibm-db2	open	syn
60000	unknown	open	syn

Responses from version scanning with not identified version

Protocol	Port number	Service	Return data
UDP	500	isakmp	SF-Port500-UDP:V=7.01%I=7%D=3/2%Time=56D6DBD7%P=x86_64-pc-linux-gnu%r(RPCCheck,24,"r\xfe\x1d\x13\0\0\0\0\0\0\0\x02\0\x01\x86\xa0\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(DNSVersionBindReq,24,"\0\x06\x01\0\0\x01\0\0\0\0\0\0\x07ver\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NBTStat,24,"\x80\xf0\0\x

			10\0\x01\0\0\0\0\0\0\x20CKA\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(SIPOptions,24,"OPTIONS\x20sip:nm\x20S\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NTPRequest,24,"\xe3\0\x04\xfa\0\x01\0\0\0\x01\0\0\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(SNMPv1public,24,"0\x82\0/\x02\x01\0\x04\x06public\xa0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(AFSVersionRequest,24,"\0\0\x03\xe7\0\0\0\0\0\0\0e\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(DNS-SD,24,"\0\0\0\0\0\x01\0\0\0\0\0\0\t_se\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(Citrix,24,"\x1e\0\x010\x02\xfd\xa8\xe3\0\0\0\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(Kerberos,24,"j\x81n0\x81k\xa1\x03\x02\x01\x05\xa2\x03\x02\x01\n\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(sybaseanywhere,24,"\x1b\0\0=\0\0\0\0\x12CONNECT\)\x20%\x20\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05")%r(NetMotionMobility,24,"\0@P\0\0\0\0\x85\]\xb4\x91\(\0\0\0\0\)\x20%\(\0\0\0\0\0\0\0\$\0\0\0\x08\0\0\0\x05"

```
Results of automated tests

Executed at 08 March 2016 04:01:01

Host info
---------

IPv4 address: 130.188.91.224
IPv6 address:
MAC address:
Vendor: None
Scan runtime: 1 hours 33 minutes and 29 seconds

Nikto test results
------------------

Nikto test results on host 130.188.91.224:8899
```

```
1 The anti-clickjacking X-Frame-Options header is not present.
2 The X-XSS-Protection header is not defined. This header can hint
to the user agent to protect against some forms of XSS
3 The site uses SSL and the Strict-Transport-Security HTTP header
is not defined.
4 The X-Content-Type-Options header is not set. This could allow
the user agent to render the content of the site in a different
fashion to the MIME type
```

```
Nikto test results for host 130.188.91.224:50000
```

```
1 Connection failed to the web server
```

Nikto test results for host 130.188.91.224:60000

1 Connection failed to the web server

Nmap scripting engine test results for the host

NSE discovery category scripts

ScriptID	Script Output
broadcast-ping	IP: 192.168.182.2 MAC: 00:50:56:f2:e4:85 Use --script-args=newtargets to add the results as targets
targets-asn	targets-asn.asn is a mandatory parameter
targets-ipv6-multicast-slaac	IP: fe80::5d32:27e5:6b3c:fb97 MAC: 00:50:56:c0:00:08 IFACE: eth0 IP: fe80::b479:9c0c:faa4:96e9 MAC: 00:50:56:c0:00:08 IFACE: eth0 Use --script-args=newtargets to add the results as targets
asn-query	BGP: 130.188.0.0/16 | Country: FI Origin AS: 565 - Technical Research Centre of Finland,FI Peer AS: 1741
dns-brute	Can't guess domain of "verifier"; use dns-brute.domain script argument.
fcrdns	PASS (224.91.vtt.fi)
hostmap-ip2hosts	hosts: Error: could not GET http://www.ip2hosts.com/csv.php?ip=130.188.91.224
hostmap-robtex	
ip-geolocation-geobytes	ERROR: Script execution failed (use -d to debug)
ip-geolocation-geoplugin	ERROR: Script execution failed (use -d to debug)
ip-geolocation-maxmind	ERROR: Script execution failed (use -d to debug)
ipidseq	Unknown
path-mtu	PMTU == 1500
qscan	PORT FAMILY MEAN (us) STDDEV LOSS (%) 22 0 812.60 53.21 0.0% 111 0 903.60 348.95 0.0% 225 1 1000758.70 4153.79 0.0% 902 0 759.20 102.40 0.0% 8899 2 1305.20 201.65 0.0% 50000 2 1318.60 340.51 0.0% 60000 3 1034.20 79.19 0.0%
whois-domain	You should provide a domain name.
whois-ip	Record found at whois.ripe.net inetnum: 130.188.0.0 - 130.188.255.255 netname: VTTNET descr: Technical Research Centre of Finland country: FI orgname: VTT Technical Research Centre of Finland organisation: ORG-VTRC1-RIPE email: info@vtt.fi

Nmap scripting engine test results for TCP ports

NSE results for TCP service ssh (OpenSSH 6.6.1 (protocol 2.0)) in port 22

See Appendix 2

NSE results for TCP service rpcbind (2-4 (RPC #100000)) in port 111

See Appendix 2

NSE results for TCP service vmware-auth (VMware Authentication Daemon 1.10 (Uses VNC, SOAP)) in port 902

See Appendix 2

NSE results for TCP service http (Tornado httpd 4.3) in port 8899

See Appendix 2

NSE results for TCP service http (Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.1e-fips)) in port 50000

See Appendix 2

NSE results for TCP service http (WSGIServer 0.1 (Python 2.7.6)) in port 60000

ScriptID	Script output
http-brute	Path "/" does not require authentication
http-chrono	Request times for /login/; avg: 209.25ms; min: 206.87ms; max: 213.36ms
http-comments-displayer	Couldn't find any comments.
http-cross-domain-policy	ERROR: Script execution failed (use -d to debug)
http-csrf	Couldn't find any CSRF vulnerabilities.
http-date	Tue, 08 Mar 2016 02:01:58 GMT; +14s from local time.
http-devframework	ERROR: Script execution failed (use -d to debug)
http-dombased-xss	Couldn't find any DOM based XSS.
http-errors	Couldn't find any error pages.
http-feed	Couldn't find any feeds.
http-fetch	Please enter the complete path of the directory to save data in.
http-fileupload-exploiter	
http-frontpage-login	false
http-headers	Date: Tue, 08 Mar 2016 02:32:45 GMT Server: WSGIServer/0.1 Python/2.7.6 X-Frame-Options: SAMEORIGIN x-xss-protection: 1; mode=block Content-Type: text/html; charset=utf-8 Location: http://verifier/login/ x-content-type-options: nosniff (Request type: GET)

http-malware-host	Host appears to be clean
http-methods	Supported Methods: GET HEAD OPTIONS
http-mobileversion-checker	No mobile version detected.
http-referer-checker	Couldn't find any cross-domain scripts.
http-server-header	WSGIServer/0.1 Python/2.7.6

http-sitemap-generator	Directory structure:
	Longest directory structure:
	  Depth: 0
	  Dir: /
	Total files found (by extension):

http-slowloris	false
http-stored-xss	Couldn't find any stored XSS vulnerabilities.
http-title	Did not follow redirect to http://verifier/login/
http-traceroute	Possible reverse proxy detected.
http-useragent-tester	Allowed User Agents: Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html) libwww lwp-trivial libcurl-agent/1.0 PHP/ Python-urllib/2.5 GT::WWW Snoopy MFC_Tear_Sample HTTP::Lite PHPCrawl URI::Fetch Zend_Http_Client http client PECL::HTTP Wget/1.13.4 (linux-gnu) WWW-Mechanize/1.34
http-vhosts	127 names had status 301
http-xssed	ERROR: Script execution failed (use -d to debug)