


Technische Universität München
Department of Informatics

Bachelor's Thesis in Informatics

Evaluation of port scan- and port scan detection tools

Simon Bauer

Technische Universität München

Department of Informatics

Bachelor's Thesis in Informatics

Evaluation of port scan- and port scan detection tools

Evaluation von Port Scannern und Port Scan Detection Software

Author	Simon Bauer
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Nadine Herold, M.Sc; Dipl-Inf Stephan - A. Posselt
Date	14.08.2015

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, 14.08.2015

Signature

Abstract

Port scans are a widely spread method to leak information about ports' status on computer systems connected to networks. Port scans are used by attackers, to prepare attacks, but are used by network administrators for, inter alia, analysis of errors, too.

Detecting port scans as early as possible, enables network and security administrators to prepare their system towards following attacks. One of the hardest problems to solve regarding port scan detection tools is the tools' ability to separate network traffic from malicious port scan traffic.

This thesis is purposed to define criteria which can be used to compare port scan software and port scan detection tools. The criteria are used to analyze different aspects of the two kinds of tools and offer certain metrics to describe the tools' quality.

Based on a scientific test environment, that is described in the thesis, different kind of port scans are performed, on which results several selected tools are analyzed. As mentioned the test environment is the basis of the performed tests of port scan- and port scan detection tools, which is hosted in the test-bed of the chair of Network Architectures and Services of the Technical University Munich.

The defined criteria are used to elevate a selection of port scan tools and port scan detection tools, which are presented and analyzed in detail. The selected port scan tools are *NMap*, *ZMap*, *Masscan* and *xProbe2*. The evaluated port scan detection tools are *Portsentry*, *PSAD*, *Scanlogd* and *Snort*'s port scan detection module *SFPortscan*.

One target of this thesis is to explain the evaluated tools' results with the used approaches of port scanning or rather the used approach of port scan detection.

Zusammenfassung

Port Scans sind eine weit verbreitete Methode, um Informationen über die Zustände der Ports eines Computersystems, welches mit einem Netzwerk verbunden ist, zu gewinnen. Port Scans können einerseits von Angreifern genutzt werden, um ihre Angriffe vorzubereiten, andererseits von Netzwerkadministratoren, unter anderem, zur Analyse von Fehlverhalten. Das frühzeitige Erkennen von Port Scan Angriffen hilft Netzwerk- und Sicherheitsadministartoren ihre Systeme auf weitere Angirffe vorzubereiten. Ein zentrales Problem beim Erkennen von Port Scans ist das Trennen von normalem Netzwerkverkehr und Verkehr, welcher zum Scannen von Ports dient.

Diese Arbeit definiert Kriterien, welche verwendet werden können, um Port Scanner und Port Scan Erkennungssoftware zu vergleichen. Die Kriterien werden verwendet, um verschiedene Aspekte der Port Scanner und Port Scan Erkennungssoftware zu vergleichen, und stellen gleichzeitig Metriken bereit, die Qualität der Programme zu beschreiben.

Auf Basis einer wissenschaftlichen Testumgebung, welche in der Arbeit beschrieben ist, wurden verschiedene Port Scan Angriffe ausgeführt, deren Ergebnisse zum Vergleich mehrerer Tools genutzt werden. Die Testumgebung, als Basis für die Test der Software, wurde im Test-Bed des Lehrstuhls für Netzarchitekturen und Netzdienste der Technischen Universität München eingerichtet.

Die definierten Kriterien und Metriken werden verwendet um eine Auswahl von Port Scannern und Port Scan Erkennungssoftware, welche vorgestellt und detailliert analysiert werden, zu evaluieren. Die ausgewählten Port Scanner sind *NMap*, *ZMap Masscan* und *xProbe2*. Auf Seiten der Port Scan Erkennungssoftware werden *Portsentry*, *PSAD*, *Scanlogd* und *Snort*s Port Scan Erkennungsmodul SFPortscan verglichen.

Ein Ziel dieser Arbeit ist es die gewonnen Ergebnisse der evaluierten Programme auf deren Ansatz zum Scannen von Ports, bzw. zum Erkennen von Port Scans zurück zu führen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The topic of network attacks is as current as never before. Daily attacks of different kinds are performed on every kind of targets. Governments, financial institutes, online shops or any web application is the target of attacks for different purposes. For example, [1] occupies with known attacks from around the world, gathering them into an attack timeline [1].

By regarding the named kinds of attacks, the targets and a description of the performed attacks, it is striking, that there is a wide range of attacking possibilities. The relevance of attackers implies the question, how attackers are able to get access to computer systems from remote.

The complexity of getting access to secure computer systems makes it necessary to prepare attacks careful. There are three parts of an attacker's step of attack preparation, according to Stuart McClure et al. [2]: the footprinting, the scanning and the enumeration.

The footprint is defined as the hacker's collection of data concerning different aspects of the target host, while the enumeration is the documentation of all gathered and leaked information relevant for the planned attack. Last but not least, scanning is a method to perform the footprinting and enumeration. Scanning means an activity of leaking information enabling the attacker to design his attack and to fulfill his enumeration. A very widely spread method of scanning is to perform port scan attacks, this thesis' main topic. Port scans are used to find open ports on one or several hosts to determine offered services, which can be exploited. Port scans allow the attacker to reproduce the target host's state and to plan an attack using the target host's weaknesses. As already mentioned, port scans are an often used method of information gathering, because of the scans' convenience to use and their performance efficiency. Port scans are based on the TCP and UDP protocols and use default implementation and processes the protocols are based on, to determine a port as open or closed.

This thesis deals with the tools used to perform port scan attacks on the one, and with tools purposed to detect port scan attacks on the other hand. Port scan detection tools are used to detect signs of attacking activities as early as possible, to allow network and security administrators to prepare their systems to be aware of following attacks.

Section 1.1 presents the research questions handled in this thesis, section 1.2 gives an overview about this thesis structure.

1.1 Research questions

This work's purpose is to describe an approach to evaluated port scan tools and port scan detection tools.

Q1: Which differences exist in the evaluated tools' quality?

Regarding the design of an approach to evaluate tools, different criteria have to be defined and described. On the one hand, it is important how the tools can be compared, on the other hand it is quite important, to ensure meaningful measurements to gather significant and comparable results.

Q2: Which criteria can be used to compare and classify the behavior and functionality of the chosen kind of software?

In case of port scan detection tools, many different aspects have to be mentioned, because the tools have an significant technical point of view, but can be seen as a user orientated software, too.

Q3: How the different criteria can be measured to compare the different tool's results?

On the one hand the metrics used to gain comparable results should be hardware independent, on the other hand the results are gathered by executed tests to be able to evaluated the results from a practical point of view, as well.

After the theoretical approach of port scan tool and port scan detection tool evaluation has been described, the metrics and methods have to be tested on existing software offerings. This step is important, to be able to assess the quality of determined comparable aspects and to assess the whole approach of evaluation.

One of the main parts of this evaluation is how the chosen approach of tool design and scan - or detection engine influences the tools' results. A possible conclusion of the practical evaluation of this software is an statement about the tools purpose and the tools' suitability for different requirements.

1.2 Chapter overview

This section introduces the thesis' structure. First the comparable aspects of both kinds of tools are described. In chapter 2 first an overview of all aspects is given, followed by a description of the criteria to evaluate port scan and port scan detection tools.

The chapter of evaluation criteria is followed by an overview of related work in chapter 3. The related work is separated in two kinds. First the related work of port scanning and port scan tools, followed by work related to port scan detection concerns is introduced.

After the related work has been presented, the tools which have been selected to be evaluated are named and described in chapter 4. The attacker tools chosen and presented are *NMap*, *Masscan*, *ZMap* and *xProbe2*. The port scan detection tools are differed in host-based and network-based port scan detection tools. The host-based port scan detection tools evaluated in this thesis are *Portsentry*, *PSAD* and *Scanlogd*. The network-based tool evaluated is *Snort* only.

After the evaluated tools are introduced, the test environment and its components are described in chapter 5. The used test-bed is presented and the used setup of virtual machines is described . The chapter of the test environment is continued by an overview of ran experiments, which were purposed to describe the port scanners behavior with results of realistic port scans.

After the test environment has been introduced, the evaluation of the selected tools begins in chapter 6. All aspects described in chapter 2 are evaluated in an own section.

Last but not least, chapter 7 consists of the most important aspects and results of this thesis and a short summary of the whole content. The appendix, chapter 8, includes different tables of raw data gained by gathering the simulation results.

Chapter 2

Criteria of evaluation

As basis of evaluating different port scan - and port scan detection software, it is necessary to choose several criteria, that can be compared. To cover a wide range of points of interest, the criteria have been chosen from different layers, beginning at the port scanners efficiency and their need of bytes per scan, up to the detectors ways of alerting attacks and output quality.

After the criteria have been chosen, it has to be defined how they are measured and which data is necessary to measure them. This chapters names the criteria which will be compared and explains how they are measured and why they are important. First, the criteria to evaluate port scan software is explained. Then the port scan detection software's criteria are described.

2.1 Criteria of port scanner evaluation

Port scan tools are mostly used by attackers of computer systems or network and security administrators. For a network administrator, there are several reasons to use port scan tools. Port scanners can be used to inventory network components like clients or servers. Next to the information which IP addresses belong to a certain device, administrators can gain information about the devices' uptime and trace routes. Port scanners can be used for the analysis of error, too. In case a tool, that needs to communicate via network, does not work as it should, an administrator can check the system for the required open ports. Another reason to use port scan tools is to uncover own weaknesses in the administrated network. By using port scan software and running scans on the own network, the administrator gets a view on security gaps and possible ways to enter the network for undesired reasons. Thanks to the analysis of port scan results, the administrator can retrace, which weaknesses his system offers for external attackers.

Another group of port scan tool users are attackers. Port scans often are the initial step

of attackers which purpose to get access to a specific hosts or network. To know which service can be exploited to leak the necessary information, the attacker can scan the host and receive the information of offered services to the target host's environment. A scan can have different purposes, therefore port scan tools offer different kinds of specifying hosts and ports.

[3] introduces different kinds of host and port specification to fulfill a scan's purpose. If an attacker is interested in attacking a certain host, he may wants to scan several port on one host, which means he performs a *vertical port scan*. If the attacker searches for a victim and already has an exploit for a certain service, he has to scan several hosts on the port, the service is offered on. This kind of port scans are called *horizontal scans*. The combination of *vertical* and *horizontal scans* are called *block scans*. *Block scans* are port scans targeting several host on more than one port. The purpose of a *block scan* performed by an attacker often is to analyze his environment and find new ways to attack hosts.

2.1.1 Scan methods

One of the most important decisions in designing port scan attacks is the selection of the used scan method. The scan method is the used technique to identify open ports and, scan technique dependent, further information. A port scanning tool that offers several scan methods gives the user the ability to create scans, that may fit better to his goals than an other method does.

TCP SYN scan—The most common scan method is the TCP SYN scan. A SYN scan tries to determine the ports' status by sending TCP SYN packets and analyzing received SYN/ACK responses. Open ports react with a SYN/ACK packet to enable the TCP *Three-Way-Handshake*. Closed ports do not react to TCP requests. The SYN scans' advantages usually are a good performance and a lower ability of being remarked compared to other scan methods.

TCP Connect scan—A more remarkable scan method than the SYN scan is the TCP Connect scan. The TCP Connect scan tries to build a TCP connection to the victims' ports to determine the ports' status. The TCP Connect scan sends TCP SYN packets, receives SYN/ACK responses from the victim and fulfills the TCP *Three-Way-Handshake* by sending a TCP ACK packet. Figure 2.1.1 shows the possible processes of a TCP Connect scan. The results of TCP SYN - and TCP Connect scans are comparable, but the TCP Connect scan can cause performance issues on the victims' systems, because the victims' systems allocate resources to enable the TCP connections. A huge number of TCP connections in a short period of time may is suspicious.

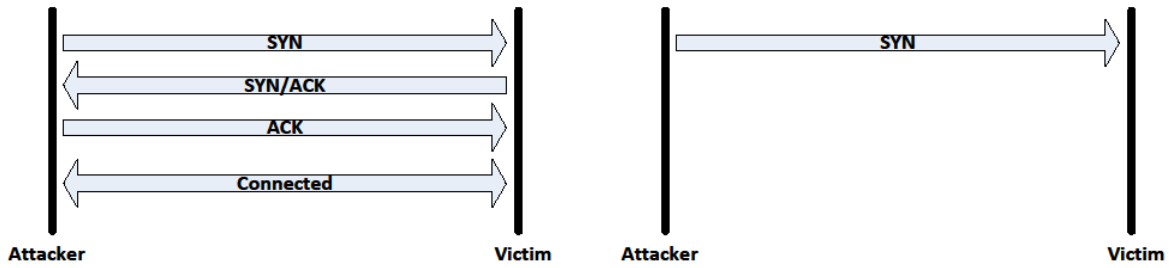


Figure 2.1: The process of a TCP Connect scan of an open port, on the left, and of a closed port, on the right.

TCP NULL -, XMAS - and FIN scan—Another way to use the TCP protocol to leak informations about remote systems are the NULL -, FIN -, and XMAS - scan. Those scan methods differ to the SYN - and TCP Connect scan. The SYN - and Connect scans try to determine the ports status by analyzing the victims behavior during the TCP *Three-Way-Handshake*, the NUlL -, FIN - and XMAS scans analyze the packets, which follow to the correspondent packets sent from the scanner to the scans' victim.

During a NULL scan the port scanner sends TCP packets, which headers' flags are all set to zero. Those TCP packets are invalid and ignored by open ports. Closed ports respond with a TCP reset packet, because the port is unreachable. The XMAS - scan exploits TCP reset packets, as well. Due to set FIN -, URG - and PUSH flags in the TCP header a port scanners is able to send another kind of invalid TCP packets.

The TCP FIN scan uses valid TCP headers. The FIN scan uses TCP packets, which headers' FIN flag is set. This kind of TCP packets is used to closed existing TCP connections. If a FIN packet is received without an existing TCP connection, the port reacts with an TCP reset packet.

Thanks to sending invalid TCP packets the NULL -, FIN - and XMAS scans are able to pass non-state-full, also known as stateless, firewalls and routers, that filter incoming packets, because they are not able to decide how to handle invalid packets. But the three scan methods that exploit TCP resets have a disadvantage, too. According to *NMap*'s scan method description [4] those scans' results are not always as significant as the TCP SYN - or TCP Connect scans' results. This is caused by different implementations of operating systems. Some operating systems, like Windows systems for example, are not implemented appropriate the RFC 793 standard [5]. This causes TCP resets as responses to invalid packets from closed and open ports, what may falsifies the scanner's results. The different implementations of operating systems enables to detect running operating

systems. This feature is explained in section 2.1.5.

TCP ACK scan—A scan method that completely differs from the other TCP scan methods is the ACK scan. The ACK scan's purpose is not to determine ports as open or closed, but to identify firewall rules and separate active firewalls in state-full and not-state-full ones. During an ACK scan the port scanner sends TCP packets, in which headers only the ACK flag is set. Ports that are not protected by a firewall react with TCP resets, ports that are protected and filtered respond with an ICMP message.

UDP scan—Another possibility to scan ports is using the User Datagram Protocol - UDP. Port scanners performing UDP scans are not able to determine open ports by analyzing the ports responses, because open ports do not react to received UDP packets. This prohibits an statement about the port's status because the port scanner's UDP packet could have been lost without reaching the scan's target. Therefore UDP scans work the other way around and only determine closed ports. Closed ports respond with an *ICMP port unreachable message*, if an UDP packet is received. This allows to determine ports that are closed. Figure 2.1.1 shows the possible processes of an UDP scan.

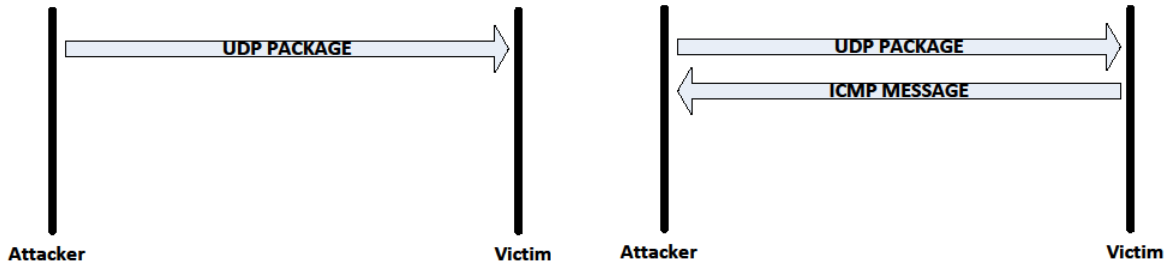


Figure 2.2: The process of a UDP scan of an open port on the left and of a closed port on the right

2.1.2 Effectiveness

The effectiveness of port scan tools is one of the most important aspects to compare port scanners quality. The effectiveness covers the results of the port scanners of certain scans, which are performed with every tools.

The effectiveness in concerned to the port status determined by the tools. On the on hand, the amount of open, closed and other port status are relevant, on the other hand, the port numbers determined as open or closed have to be the right ones. A port scanner

port status	determined status	classified as
open	open	correct
open	closed	false
closed	open	false
closed	closed	correct
	no status determined	undefined

Table 2.1: Overview of the classification of port scanners' results

with a low effectiveness is useless, because port scans are purposed to leak information. The output of false information does not bring a benefit to the tools' users.

The effectiveness of port scan tools is separated into the *vertical scans* and the *block - and horizontal scans*. This is necessary, because block and horizontal scans can be analyzed for the amount of scanned hosts, beside the amount of determined open ports and the determined open port numbers.

Table 2.1.2 introduces how the port scanners' results can be classified.

2.1.3 Efficiency

The efficiency is a criteria that influences two points of interest the executor of a port scan may has. Attackers and administrators are interested in getting fast results and attackers do not want to be detected. Both aspects are influenced by a port scanner's efficiency.

Time efficiency—That faster a port scanner fulfills scans, that faster the scan executor gets his result. The scan duration of a port scan tool is dependent by at least two variables. First, the data rate the port scanner is able to perform and, second, the port scan's configuration and used method. Additional the port scanner's time efficiency depends of the used hardware, too. Slower computers may are not able to handle a fast scan rate, even if the port scanner would be able to. For that reason the measured scan times are not compared as absolute values, but as indicators for a port scanners scan speed in relation to the other tools' speed.

Data efficiency—But scan efficiency is not limited to the time a scan needs. Another point of view are the sent and received data packets and the related data size. Stuart Staniford et al. [6] define the following metrics to determine a port scans size.

The may most intuitive metric to determine a scan's size, according to Staniford et all., is the total scan size. The total scan size is the amount of all scanned IP/port combinations. Another metric is the total information size of a scan. This metric considers the data size

necessary to send or receive data to gain additional information, like OS fingerprints. The authors name the closed scan size as a metric to describe a scan's conspicuousness. The closed scan size is based on the fact, that scans of closed ports are much more conspicuous, than scans of open ports, because it is less probable, that normal network traffic contacts a closed port.

2.1.4 Randomization

The randomization of subnet hits, hosts within a subnet and the order of scanned ports on a single host, can influence the risk of getting detected clearly in different ways.

Big *horizontal* - and *block scans* can cover several subnets. In case network intrusion detection systems monitor whole subnets, a randomization of subnet hits is important. A scanner that penetrates one subnet after another performs more suspicious scans than a scanner, which randomizes his probes between the subnets and creates an realistic distribution.

Scans that cover several hosts have a great potential to benefit from a port scanner's randomization ability. The randomization of hosts can help to avoid the detection by network intrusion detection systems and host-based port scan detection tools.

The randomization of port order disables detection tools to discover patterns in received traffic and makes the scan probes look more like real traffic. The randomization of an order can not be proofed scientific.

Therefore, the evaluation of the port scanners' ability to randomize is based on observation during the tests and the analysis of different probes, targeting to visualize the results and search for a higher concentration of host or port hits. To analyze the distribution of scan probes on different subnets, scans have to be performed which targets are subnets.The subnets have to include the same amount of hosts, to make sure that the probability to hit a specific subnet is equal to the probability to hit one of the other subnets.

Similar to the method of analyzing the hit distribution of different subnets, the randomization of host order is based on scans targeting the hosts directly by their IP addresses. After the scan is performed the order of scanned hosts corresponds to the order of sent packets to the target hosts.

Last but not least, the analysis of randomization of ports on a single host is based on the order of contacted ports by the port scanner on the victim's side. Comparable to the host order it is deciding in which order the first contact took place. This is important, because of the undefined order of packets following to the initial ones. For example during a TCP Connect scan the order of sent SYN packets is the deciding one and not the order of SYN/ACK and ACK packets.

2.1.5 Configuration of scans and OS detection capabilities

Nowadays, port scanning software offer more capabilities to leak information about the scan victim's system, beside the ports' status. Further information can be trace routes, the systems' up time or the explicit reason for the port scanners status determination. Other possibilities to configure scans are a fixed data rate, set by the user, or waiting periods between different scan probes, to stay less suspicious. The capabilities of the evaluated tools are compared and analyzed in their amount and usefulness.

As explained in section 2.1.1, port scanners use different scan methods, to leak as much information as possible. Several tools offer a operating system detection function, which is performed during the port scan. This function may is useful for an attacker or system administrator, to get an overview over a network and its components.

2.2 Criteria of port scan detection tool evaluation

Port scan detection tools are used to detect port scan attacks. Usually port scan detection tools are used by network administrators to monitor clients and servers and to detect port scan attacks. From a network administrator's point of view, port scan detection tool are helpful to be aware of port scan attacks without need of time after the tools' integration.

2.2.1 Reliability

The most important aspect of port scan detection tools is the reliability of scan detection. A port scan detection tool, that does not alert port scans reliable, may is not the best way to protect hosts and networks. The reliability of port scan detection tools is evaluated by four metrics, which cover all possibilities of alerts and scans which are not alerted.

False - positives—The first metric to describe a detection tool's reliability are false - positive. False - positives are defined as scans that have been alerted without a port scan has been performed. This means the detection tool is not able to separate normal network traffic from port scan traffic, which is a huge disadvantage for users.

True - negatives—The next metric is called true - negatives. True - negatives are defined as probes of port scan detection tool log files, without alerted port scans and no port scan performed in a specified period of time. This makes the true - negatives more harder to analyze than the false - positives, because false - positives are an indicator for false behavior of the port scan detection tool, while true - negatives are a part of the expected behavior of a port scan detection tool, as far as no scan is performed.

scan performed	scan alerted	classified as
✓	✓	true - positive
✓	✗	false - negative
✗	✓	false - positive
✗	✗	true - negative

Table 2.2: Overview of the four metrics, that describe a detection tool's reliability

False - negatives—Similar to the false - positives, the false - negatives are easier to remark. The false negatives are defined as port scans that have been performed, but have not been alerted by the port scan detection tools. This means the detector does not identify a port scan attack, while a port scan was targeting the host. This is may the worst case for the reliability of detection tools, because false - negatives mean, that the user is not informed about an incoming danger.

True - positives—True - positives are the last metric, which describes the detection tools' reliability. True - positives occur, if a port scan is performed and the port scan detection tools alert an attack. Similar to the true - negatives, true - positives are expected behavior of a port scan detection tool. The true - positive frequency of a tool is determined by performing port scans and look for alerts for the performed scan. A port scan detection tool with a higher frequency of true - positives has a better reliability then a tool with a lower true - positive frequency.

Table 2.2 summarizes the four kinds of reliability classification.

2.2.2 Anti-scan reaction

Beside the ability to detect scans, some port scan detection tools offer measurements to react to port scan attacks. This aspect compares the quantity of anti-scan reactions the detectors offer and the effectiveness of the different measures.

During the tests, two points of interest are observed. First, as mentioned, the quantity of offered measures. Second, the impact the anti-scan reaction causes. To analyze this criteria it is important to understand the approach the reaction follows, which impact it theoretically has and how well the tool is able to achieve the expectations the reactions promise.

In this criteria the tools may differ a lot, because it is not a part of the port scan detection tools' definition to react on detected scans. Therefore, the different tools could offer from none reactions up to plenty of measures. In this point the analysis of the anti-scan reactions is correlated to the analysis of the detection tools' reliability. A tool which alerts plenty of false - positives, may prohibits any communication between, the falsely

identified , attacking host and the host the detection tool is running on, without any need or danger.

2.2.3 Usability

The next criteria to compare port scan detection tools is the tools' usability. The usability is defined as a combination of the tools' documentation, the ways the tools can be installed and the complexity of the tools' integration to a network or set them up on a single host. Because those points of interest can not be compared scientifically, the observations made during the setup of the test environment are described and noticeable problems and features are explained.

The analysis of documentation is based on the official documentation files, websites and commented configuration files offered by the port scan detection tools. It has to be mentioned, that this analysis only covers documents published by the tools developers or enterprises and does not consider any other helpful sources such as web-blogs, forum entries or any other way of explanation and documentation, created by other users. On the first glance the analysis of how the tools can be installed and how they are successful integrated may similar, but these are to different steps on the way to a running port scan detection tool. This is caused by the detectors dependencies to other system components and software. This often makes it more complicated to integrate the tools after their installation.

2.2.4 Output quality and format

Another aspect to compare port scan detection tools is the output a tool generates. On the one hand the output has a certain quality based on included information and the structure port scan alerts are presented in.

This criteria is may more interesting as it seems to be. For example a port scan is performed and scans 100 ports on one host. The detector detects the scan and creates a port scan alert in the tool's specified log - or alert-file. Before the alerts are created single scans of single ports have to be detected. This includes, that port scan detection tool is not able to group single scans to one big scan in the first moment. After the single scans have been detected, some tools may are able to group the detected scans to one big scan including the whole port range of the 100 ports instead of alerting 100 single ports.

A tool with the ability to group single scans, what makes the output much more readable for humans, has an advantage towards tools alerting every single scan. Next to the ability of grouping scans the right way, the given information for detected scans is important. Tools may output information like the scanned port range, the used scan

type, the attacker's IP and other information helping the user to understand the detected attack.

The format of the given output is important, too. The way the output is represented, decides about how the detection tool's results can be handled automatically. This may is an aspect of interest, which may is more relevant to administrators of bigger networks, who have to monitor a lot of hosts at the same time and are not able to keep all single hosts in mind at the same time. Another aspect of the output format is the method used to inform the user about the detected scans. Next to log-files for scan alerts, terminal commands to present gathered results or even e-mails are conceivable.

2.2.5 Configuration

The usage of port scan detection tools in many different situations, makes it necessary to configure port scan detection tools referred to the users' needs. The first point of interest is to get an overview of all configurable options each tool offers, to compare the quantity of configuration possibilities.

In general examples for configurable options could be the monitored port range, options to configure alerts or different danger level, which help the user to categorize port scan attacks.

After the quantity of configurable options are determined, different configurations have to be created, tested and compared to the results of the default configuration. This shows how the port scan detection tool can be influenced by their configuration and how different configurations handle certain port scan attacks. To get significant results it is meaningful to create a configuration, which is more aware of port scans and another configuration, which is less aware. This makes it more easier to compare the results of different port scan detection tool configurations.

The configuration of the tools is an aspect correlated to the aspect of usability of port scan detection tools, because an appropriate documentation of all options makes it more easier to create better adapted configurations.

2.2.6 Approach of port scan detection

The approach a detector uses can be compared to other approaches. This criteria is more theoretical, then the aspects explained above, because its evaluation is not based on experiments' results or observation but on the analysis of behavior of each tool, supported by the analysis of source code and developers' information.

The approaches may can explain the results of other criteria and gives an overview of possible approaches and their advantages or disadvantages. During the other criteria are

used to evaluate the tools itself, this one is purposed to inform and analyze theoretical ideas of port scan detection. Combined with the results of tested and approved data, expectations of theoretical approaches can be confirmed or negatived. The theoretical approaches of each tools are explained in chapter 6.

2.3 Overview of all evaluated aspects

Table 2.3 gives an overview of all criteria explained and evaluated in chapter 6. The first column includes the criterion ID, shortened CID, which is purposed to identify the different criteria and to give an reference between the evaluation and this chapter. The second column names the criteria correlated to the criterion ID from the first column.

CID	Criterion
Port scanner criteria	
S.I	Scan Methods
S.II	Effectiveness
S,III	Efficiency
S.IV	Randomization
S.V	Configuration of scans and OS detection capabilities
Port scan detection tool criteria	
D.I	Reliability
D.II	Anti-scan reaction
D.III	Usability
D.IV	Output quality and format
D.V	Configuration of detection
D.VI	Approach of port scan detection

Table 2.3: Overview of all criteria evaluated

Chapter 3

Related work

Port scans are an essential way step to prepare attacks or monitor networks. This is the reason for many researchers to deal with port scans as subject of research. For this thesis several kinds of related work are important. The first topic of related work is the evaluation of port scan tools. The second one the evaluation of port scan detection tools and their theoretical approach to detect port scans.

3.1 Work related to port scan tools

The 2015 published thesis of Nils Mäurer [7] deals with a comparison of the efficiency of the port scan tools *NMap*, *ZMap* and *Masscan*. The author explains the problem of network administrators, who have to deal with big networks. For example a scan of the whole *Münchener Wissenschaftsnetzwerk* with *NMap* needs up to six month. Mäurer's result towards the scanners effectiveness is, that the scanners do not differ a lot. As a solution for the requirements of the MWN Mäurer introduces an own port scan tool based on *NMap* and *Masscan*.

Daimi and El-Nazeer evaluate eight port scan tools using 15 criteria in [8]. The criteria refer to the options to configure scans the tools offer and do not refer to the tools' characteristics like the effectiveness or efficiency. The work informs about which port scan tools are available and which options they offer. The results of the evaluation are presented as tables without an description of the test environment and setup. The authors compare the advantages of each tools from an administrator's point of view. This differs the authors' work from this thesis, because this thesis deals with the influence of port scan options on the risk of getting detected and with the reasons for the tools behavior.

A more theoretical approach to occupy with port scanning is the analysis and review of port scanning techniques, as done by Marco de Vivo et al. in [9]. The paper deals with

port scanning techniques and methods like the SYN, FIN or ACK scan. Beside the scan techniques, the authors describe several methods of stealthy scanning, proxy scanning and other scanning techniques.

3.2 Work related to port scan detection tools

Lee et al. describe an approach to categorize port scan attacks. The authors wish to help to "generate better network intrusion detection systems" [3]. As a most distributed approach of port scan detection tools the authors mention the approach to detect scans, by analyzing the amount of connections from one source in a given time period. As examples for more complex approaches the authors name BRO IDS and Spice. According to Lee et al. port scans can be categorized by the size of scans, the scan duration and in *vertical scans, horizontal scans* and *block scans*. The differentiation in *vertical -, horizontal* and *block scans* is adapted in this thesis, because different port scan detection tools may have advantages by detecting on of the three scan types.

In his project report [10] Ruili Geng summarizes scan techniques and scan types as SYN scans, UDP scans and many more. Beside the introduction to port scan techniques the author informs about several approaches to detect port scans and explains their advantages. The author describes his own approach to detect port scans, which is similar to *Snort* one's. He also names several port scan detection tools and describes their approaches superficial. Last but not least Geng mentions, that he did not had the time to compare the results of the mentioned port scan detection tools. This differs his report to this thesis, because the evaluation of port scan detection tools is a main part of the thesis.

Comparable to this work's approach of analyzing port scan detection tools, Simon Biles investigates a port scan detection product called SPADE [11] in detail, too. Biles describes SPADE's history, concept, function and the tool's set up in-depth. Unfortunately there is no running version of SPADE for the current *Snort* version and no binaries for older *Snort* releases have been found. In difference to Biles' work, this thesis compares different approaches of port scan detection tools and does not focus on one single detection product.

Beside the analysis of theoretical approaches and the evaluation of different tools, the improvement of the detection tools reliability is a point of interest, too. For example, Y. Chabchoub analyze the possibilities of "improving the detection on on-line vertical port scan in IP traffic" [12]. This kind of related work combines theoretical approaches of port scan detection with the implementation and improvement of port scan detection engines.

Comparable to Biles' analysis of SPADE, Annie George et al. describe an approach of

a Snort preprocessor using a vector machine algorithm [13]. In general algorithms of intrusion detection systems can be related to approaches of port scan detection.

Another approach of surveying port scans is to analyze the challenges and possibilities to detect slow and stealthy port scans. Dabbagh et al. describe slow port scans as a possibility to bypass most port scan detection tools in [14]. The authors worked on a method to detect slow port scans more effective.

Chapter 4

Tools

This chapter includes a presentation of port scan and port scan detection tools evaluated in this thesis. First the port scan software is introduced, followed by the introduction of port scan detection tools.

4.1 Attacker tools

The offer of public domain port scan software is numerous. The selected tools *NMap*, *ZMap*, *Masscan* and *xProbe2* are selected for particular reasons, which are explained in the following topic. The evaluation's results of the tools are presented in chapter 6.

NMap—*NMap* is one of the most popular port scan and network monitoring tools. It is known for its large offer of capabilities to configure scans, a big variety of scan methods and a detailed documentation of all functions. Published in 2001 *NMap* is available for Linux, Windows and Mac operating systems. To make scanning more easier, users can use *ZNNMap*, a GUI supported extension of *NMap*, according to *www.nmap.org* [15]. *NMap* enables *vertical -*, *horizontal -* and *block scans* thanks to flexible implementable command line input. *NMap* commands accept several ports listed comma-separated or given as port range. The scan's targets can be parametrized as listed IP addresses or whole subnets. To view all capabilities *NMap* offers, consider the evaluation of configuration and capabilities in chapter 6. *NMap* has been updated and improved regular, therefore *NMap* is expected to be able to cover most requirements of a port scan tool.

Masscan—According to its publisher Robert Graham, *Masscan*'s purpose are fast port scans of large host and port ranges. [16] *Masscan*'s capabilities are not as extensive as *NMap*'s, caused by its focus on scan speed and efficiency. *Masscan* is available and

designed for Linux systems, but according to its developer Robert Graham [16] the code runs on other systems, too.

Graham reports of a scan performed to scan the whole Internet, which has been finished within three minutes. Thanks to a special driver, that transfers packets directly to the network hardware and mutex-free thread synchronization, *Masscan* is able to reach a packet rate of 25 million packets per seconds. Comparable to *NMap*, *Masscan* enables *vertical -*, *horizontal -* and *block scans*.

Masscan's development team calls the algorithm for randomized scans a "major feature of async port scanners" [16]. The algorithm chooses an IP address from the given addresses independent of the scanned port, which is selected by using a calculation including modulo. The algorithm of *Masscan*'s randomization is explained in detail in [17].

ZMap—*ZMap* is designated as enabling "researchers to easily perform Internet-wide network studies" [18] and its purpose are fast and large scans. On the contrary to the other tools, *ZMap* only offers *horizontal scans*, because only single ports can be specified as targets. *ZMap* offers the possibilities to scan whole subnets and multiple IP addresses.

To avoid overhead incurred by the kernel from tracking open TCP connections, *ZMap* sends packets at the system's Ethernet layer. On *ZMap*'s homepage an elaborate documentation [19] can be found, which helps to understand *ZMap*'s' functions and usage.

xProbe2—In the evaluation of port scan tools *xProbe2* may is the odd one out. Whereas *NMap*, *Masscan* and *ZMap* are purposed to scan ports and may get additional information about remote systems, *xProbe2*'s purpose is the fingerprinting of remote operating systems, according to *xProbe2*'s homepage [20]. Table 4.1 includes an overview of all fingerprinting modules provided by *xProbe2*.

Next to its OS fingerprinting function, *xProbe2* offers a port scan module for TCP and UDP scans. The port scan modules do not offer scans of multiple target addresses, so it is necessary to iterate *xProbe2* commands to simulate *horizontal -* and *block scans*.

xProbe2 has been selected to compare the advantages of port scanning tools towards port scan modules as additional information leak and to compare the results of *NMap*'s OS fingerprint quality.

4.2 Port scan detection tools

In difference to the port scan tools, Linux-based port scan detection tools are not as numerous available as the tools purposed to perform port scan attacks. Port scan detec-

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 calculation
5	infogather:portscan	TCP and UDP PortScanner
6	fingerprint:icmp_echo	ICMP Echo request fingerprinting module
7	fingerprint:icmp_tstamp	ICMP Timestamp request fingerprinting module
8	fingerprint:icmp_amask	ICMP Address mask request fingerprinting module
9	fingerprint:icmp_port_unreach	ICMP port unreachable fingerprinting module
10	fingerprint:tcp_hshake	TCP Handshake fingerprinting module
11	fingerprint:tcp_rst	TCP RST fingerprinting module
12	fingerprint:smb	SMB fingerprinting module
13	fingerprint:snmp	SNMPv2c fingerprinting module

Table 4.1: Overview of fingerprintings modules provided by *xProbe2*

tion modules often are included in intrusion detection systems by default. Nevertheless, four different tools have been selected. The tools are separated into host-based and network-based port scan detection tools.

4.2.1 Host-based port scan detection tools

One way to detect port scans is to analyze the incoming and outgoing packets of one single host. Port scans may cause an big amount of packets depending on the port range and the used scan technique. The host-based tools' disadvantage is their weakness to detect horizontal scans. If only a single port is scanned, the characterization of traffic in port scan traffic and normal network traffic gets harder.

Portsentry—The first host-based port scan detection, which will be evaluated is *Portsentry*. *Portsentry* monitors TCP and UDP ports to identify and alert scans. *Portsentry* is described and can be downloaded on its website [21]. *Portsentry* is able to differ a lot of scans like TCP Connect-, SYN-, FIN-, NULL-, XMAS-, FULL XMAS- and UDP scans, according to [22]. Even not identified scans are alerted.

Beside alerting *Portsentry* is able to start measurements against the attacker. The measurements can be specified in the configuration file. Inter alia, *Portsentry* can be configured to add TCP wrapper rules to the system's `host.deny` - files. Which does not influence the attackers results but prevents the attacker to connect to any services.

PSAD—Comparable to *Portsentry*'s possibilities to be configured, are *PSAD*'s capabilities of being configured. According to [23], *PSAD* consists of three daemons, which use logged messages from the firewall. *PSAD* analyses TCP header flags to determine the detected scan's used scan technique.

Beside analyzing traffic for port scans, *PSAD* offers additional security measurements. *PSAD* analyses different TCP, UDP and ICMP signatures to detect suspicious traffic, as probes for backdoors, DDoS- or OS fingerprinting attacks. Another feature *PSAD* offers is to try to passively fingerprint the attacker's OS system by analyzing TCP SYN packets.

Scanlogd—The third analyzed port scan detection tool is *Scanlogd*. *Scanlogd*'s approach to detect port scan attacks is to separate ports into privileged and non privileged ports, to measure, how many packets have been received in three seconds. [24]

Scanlogd is available for Linux systems and requieres the libraries `libnet` and `libnids` .

4.2.2 Network-based port scan detection tools

Another approach to identify port scan traffic, is to monitor whole subnets or specified collections of hosts. Tools that offer the possibility to detect port scans on several hosts by analyzing captured network traffic are called network-based port scan detection tool. The idea of network-based tools has an essential advantage towards the approach of host based detection tools: Network-based tools may are able to detect horizontal scan, too.

Snort—One possibility to detect port scans with a network-based detection tool is the very well-known intrusion detection system *Snort*. *Snort* is available on [25], where it is available as free download. *Snort* includes a port scan detection module called *SFPortscan* by default. In the following *Snort* will be named as the port scan detection tool and not *SFPortscan*, because *SFPortscan* is a included module only.

Snort is able to detect different classes, which are explained in the module's documentation website [26] in detail. *Snort* classifies port scans into traditional scans, distributed scans and port sweeps. *Snort* defines traditional port scans as scans, which are performed by one attacking host targeting a single host only. This way, vertical scans are defined, too. Distributed port scans are defined as scans of many hosts towards one target. The advantage to distribute a scan's workload to different attacking hosts, is to confuse the detection tool or to by pass port scan detection tools' port scan detection engines.

Last but not least, port sweeps are defined as scans performed by one attacking host, which scans several target at the same time. This definition combines the definition of horizontal and block scans. Beside the classification into different types of scans, *Snort* differs scans into IP, UDP and TCP scans. *Snort* offers a wide range of configuration possibilities. For example, the alerted protocol types can be specified. Next to the TCP-, UDP- and IP- protocol, *Snort* can be configured to watch for ICMP messages or all protocol types at the same time.

Furthermore it can be configured, if traditional, distributed, port sweeps or all types of scans are alerted and a sense level can be specified as low, mid or high. A very special feature of *Snort* is the detection of ACK scans.

It has to be mentioned, that *Snort* offered and enabled other port scan detection modules, too. For example SPADE has been a preprocessor for *Snort*, but was not maintained according to Biles [11].

Chapter 5

Test environment

5.1 Testbed structure

The chair for network architectures and services of the Technical University Munich runs a test-bed for the simulation of network attacks, detection tools and other network based researches. The test-bed offers the possibility to create own network structures and simulate traffic in a realistic environment.

The test-bed provides 2 physical machines each providing 2 subnets. Therefore, in total four different subnets can be used. Figure 5.1 shows the 2 physical machines called *Host1* and *Host2*. Beside the hosts, the routers, the subnets and the controlling network can be seen.

For a research network, where network traffic is analyzed in detail, it is important to separate network traffic caused by the running simulations from the traffic, which is necessary to communicate with the virtual machines. To achieve this separation, it exists a second network within the test-bed called *controlling network*. The *controlling network* enables the communication with the virtual machines, without falsifying the simulations' and test's results.

For the evaluation of port scan - and port scan detection tools, both hosts are used. One for the target hosts, the other one to perform port scans from external hosts.

5.2 Setup of virtual machines within the test-bed

To ensure a scientific test environment, it is necessary to structure hosts inside the test-bed in a manner, that guarantees the same conditions for all hosts and tools. Without an environment ensuring the same conditions and properties to all environment

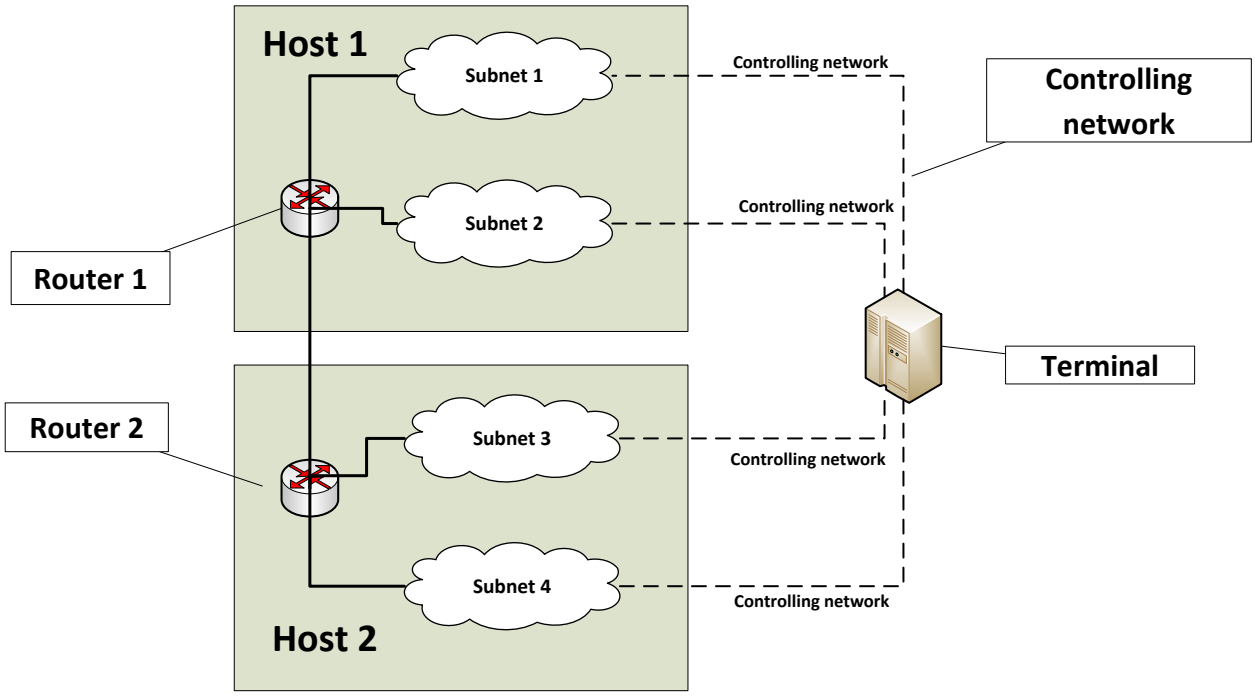


Figure 5.1: Structure of the chair's test-bed

components, the evaluations results can not be compared and evaluated in a meaningful way.

Inside the used test-bed, all hosts are virtual machines, created with the same hardware resources and using Linux Ubuntu 14.10. The advantage of using virtual machines in on the one hand a flexible structure of the host which are part of the evaluation. On the other hand virtual machines represent a more cost-efficient way, than real hardware-based hosts. Beside the decision of using virtual machines, it is important to create several VMs with the same purpose to be able to run tests on different machines to ensure machine independent runs and results of the same test.

Figure 5.2 shows how the chair's test-bed has been used to build an own test environment to run port scanners and port scan detection software. The host named *Network Monitoring* is necessary to run network-based port scan detection tools, like *Snort*, because the tools require a network monitoring port. The boxes include the installed tools on the certain VMs.

Especially the machine independent results are an important aspect for evaluating software. The behavior of software can be dependent on the hardware or virtual hardware resources the host is able to use. As already described in chapter 2, the criteria to compare port scanners and port scan detection tools are selected as hardware independent

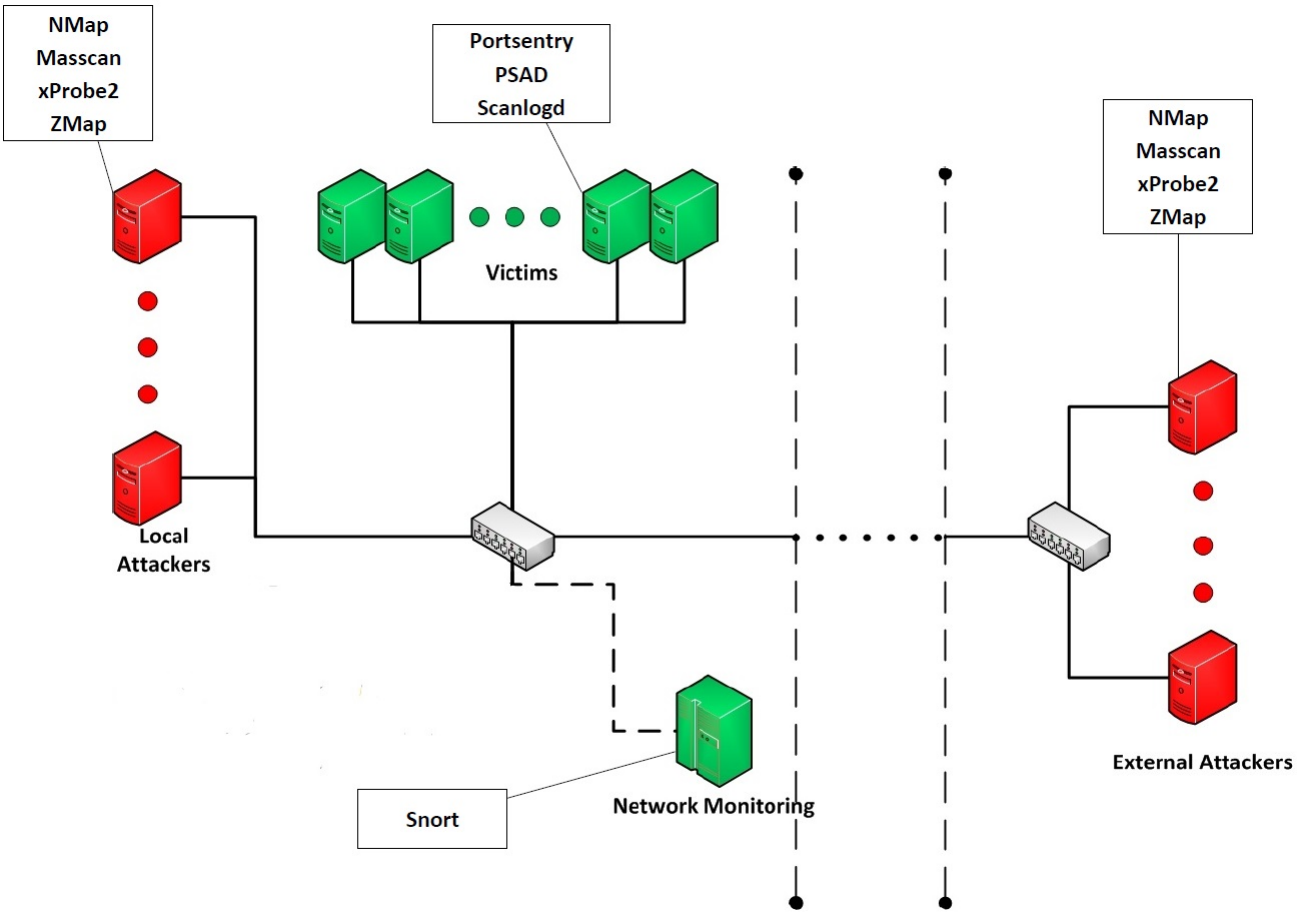


Figure 5.2: Setup of the used hosts in the chair's test-bed

as possible.

The used virtual machines are differed in two groups of hosts. The first group is purposed to perform port scan attacks towards the second group, which is purposed to run port scan detection tools. In the following the two kinds of virtual machines are described more detailed.

5.2.1 VMs running port scan tools

The virtual machines performing the port scan attacks, are called attacker VMs. The attacker VMs are separated into local attackers and attackers from outside the target's local are network and therefore are named external attackers. The separation of attacking hosts is purposed to enable an analysis targeting to show, if the port scanners' or port scan detection tools' results are dependent of the network location.

The virtual attacker machines are running Linux Ubuntu 14.10 and are equipped with every port scan tool evaluated in this thesis. The virtual machines have a dynamically allocated main memory, which is limited between 512 MB up to 4 GB, depending on the VMs resource demand. Because the VMs' operating systems are configured to not use graphical output the resource demand is manageable. The disclaimer of graphical output makes it necessary to control the hosts via the Linux terminal.

5.2.2 VMs running port scan detection tools

Next to the attacking VMs, virtual machines are necessary, which are the targets of the performed port scan attacks. Those VMs are called victims and are separated in their purpose. On the one hand victim VMs are used to analyze the port scan detection tools in detail. Those host are called full victim clients. The full-victim clients are setup with Ubuntu 14.10 as described above for the attacker VMs.

The full victim clients are supposed to be the victim of the vertical, horizontal and block scans.Therefore all host-based port scan detection tools are installed on them. *Snort* is not installed on one of the victim VMs, because it runs on hosts supposed for network intrusion detection systems and monitors the network traffic via a monitoring port.

The other kind of port scan targets are the cheap victim clients. The cheap victims are configured similar to the full victim clients, but are not used for the evaluation of port scan detection tools' result, but to enable enough targets for the analysis of horizontal, block and distributed block scans.

5.3 Experiments

To collect the necessary data to evaluate the port scanners' results and to trigger port scan detection alerts, several types of experiments are used. On the one hand, realistic designed scans have to be performed to see how port scanners behave in their proposed domain. On the other hand, scans are designed to evaluate particular criteria.

Table 5.1 shows 14 scans, separated in eight *vertical scans* (V), three *horizontal scans* (H) and three *block scans* (B), which simulate potential real scans from an attacker's or network administrator's point of view. This scans are used later to compare the tools effectiveness and time efficiency, as described in section 6.1. It has to be mentioned that the scans were performed wit all tools offering the necessary scan types. The table's first column presents the scan ID, abbreviated SID, which helps to identify the scans in the following description and analysis. Next to the scan ID the table informs about the scan's categorization in horizontal -, vertical - and block scans, the port scan tools able to perform the specified scan, the amount of targets and the port range the scan covers. The column on the right shows the used scanning method.

The scans specified in table 5.1 followed different purposes. The *vertical scans* show which tools are able to scan several ports on one host and inform about how many information the port scanners are able to leak and how they are written to the scans output.

The *horizontal scans*, comparable to the vertical scans, show which tools are able to target several hosts. Furthermore, they show how port scanners organize their result output. The *block scans* are as known a combination of *vertical* - and *horizontal scans*.

Beside the amount of determined open, closed or filtered ports, it is a very important aspect how tools inform the user about the results of such huge scans. For example a network administrator wants to inform himself about the status of all his hosts in a class C network. Class C network include up to 254 active hosts. Scanning a whole class C network means to scan 254 times 65535 ports. This means the scanner's output has to inform about 16.645.890 ports' status. Of course a lot of those ports may are closed, but the fact that huge scans are a challenge to be outputted as well, should not be forgotten.

As seen in the table, some scans target a port range named *selected ports*. *Selected ports* are defined as those ports often in use or reserved to a particular service. The amount of *selected ports* is 47. The ports are listed in table 8.1 in the annex. This idea is to simulate scans targeting a realistic port range.

SID	Category	Port scanners	Targets	Ports	Scan method
1	V	*NMap*; *xProbe2*	one	all	TCP Connect
2	V	*NMap*; *Masscan*; *xProbe2*	one	all	SYN
3	V	*NMap*; *xProbe2*	one	all	UDP
4	V	*NMap*; *Masscan*; *ZMap*; *xProbe2*	one	selected ports	SYN
5	V	*NMap*	one	all	ACK
6	V	*NMap*	one	all	XMAS
7	V	*NMap*	one	all	NULL
8	V	*NMap*	one	all	FIN
9	B	*NMap*; *Masscan*; *xProbe2*	all	all	SYN
10	B	*NMap*; *Masscan*; *xProbe2*	all	selected ports	SYN
11	B	*NMap*	all	selected ports	TCP Connect
12	H	*NMap*	all	one	TCP Connect
13	H	*NMap*; *Masscan*; *ZMap*	all	one	SYN
14	H	*xProbe2*; *NMap*; *ZMap*	all	one	UDP

Table 5.1: Overview of performed scans, purposed to simulate realistic port scans

Chapter 6

Evaluation of port scan and port scan detection tools

The evaluation is based on the defined criteria in chapter 2 and covers every single aspect which will is compared in this thesis. To be able to evaluate the different aspects, aspect-specific experiments have been performed and observations during the whole evaluation have been documented to be able to give an as detailed as possible impression of the different tools quality. Because not all aspects could be measured with metrics giving back characteristic numbers, it was necessary to observe the tools' behavior in details, too. The problem of aspects based on observation, are possible subjective influences. Therefore the aspects compared based on observation are described as objective as possible and do not include an assessment of the tools quality but only the observed behavior. After the evaluation of port scan tools and port scan detection tools, section **??** includes an overview of the most important results.

6.1 Evaluation of port scan tools

This section includes the evaluation of all port scan detection tools presented in section 4.1, considering all criteria described in section 2.1.

6.1.1 Scan methods (S.I)

As described in section 2.1.1 different scan methods help the user to adapt scans to his needs. As shown in 6.1 *NMap* offers the greatest amount of different scan methods. The reasons for *NMap*s flexibility are the well-known developer Gordon Lyon also known as Fyodor and a long history of enhancements since 2001. Lyon also took part in the development of xProbe and has a lot of experience within the range of network scanning software. *Masscan* only offers TCP SYN scan, this limitation of scan method

may is caused by the purposed high scan speed, which is dependent on the scan type. *ZMap* and xProbe offer each one TCP and UDP method. The method xProbe uses is not specified, but an analysis of sent TCP packets determined the method as TCP syn scan.

Tool	SYN	Connect	UDP	ACK	FIN	XMAS	NULL
NMap	✓	✓	✓	✓	✓	✓	✓
Masscan	✓	✗	✗	✗	✗	✗	✗
ZMap	✓	✗	✓	✗	✗	✗	✗
xProbe2	✓	✗	✓	✗	✗	✗	✗

Table 6.1: Overview of available scan methods

6.1.2 Effectiveness (S.II)

The evaluation of the port scanners' effectiveness is based on scan results of scans, which have been performed with a big host and port range. Table 5.1 shows the performed scans covering horizontal, vertical and block scans. While the scans' duration has been used to analyze the port scanners' time efficiency in section 6.1.3, the determined port status are used to analyze how many ports are open, closed or filtered and which ports were determined to a certain status.

The results of the scans performed for the evaluation of effectiveness can be found in the appendix in the table 8.2 up to 8.12. The raw results are correlated to the scans described in section 5.3.

Setting—The setting of the test environment has to be constant, to ensure comparable results performed with the same conditions. All used hosts offered the same services and all hosts has opened the same ports. On each host 48 ports were open and all 24 hosts were up while the tests have been performed.

The knowledge about every port's status enables to compare the tools' effectiveness by true, false and undefined determined port status. A true determined port status means, that the port was determined as open and the port has been open. The same way for closed ports. False determined port status are ports determined as closed or open, while the port was open or closed, in this case the port scanner failed to determine the right port status. Third, the port's status can be determined as neither closed or open, but as filtered, unfiltered or unreachable. In this case the result is handled as undefined.

Result summary—In case of the tools' effectiveness towards vertical TCP scans, the tools' quality is on a high level. Most scans performed with *NMap*, *Masscan* and *xProbe2* determined the right amount of open and closed ports and the correct port numbers of

open or closed ports. The UDP scan performed with *NMap* and *xProbe2* was not able to reach the high level of the tools' TCP scan effectiveness.

The block scans evaluated resulted in a very high level of effectiveness, too. In case of the *TCP SYN scan* all tools determined the correct 48 open ports. This means all scanned 1.572.840 ports have been determined correctly.

The tools effectiveness in case of horizontal port scans is not as high as the the tools effectiveness towards vertical or block scans. The tools partly were not able to reach all hosts specified as targets, while the hosts were up. This implies a higher rate of undefined port scanner results.

Figure 6.1 presents the tools' effectiveness in case of the scans correlated to SID 2,4 and 13, as described in table 5.1. The figure separates ports determined correct, incorrect and undefined.

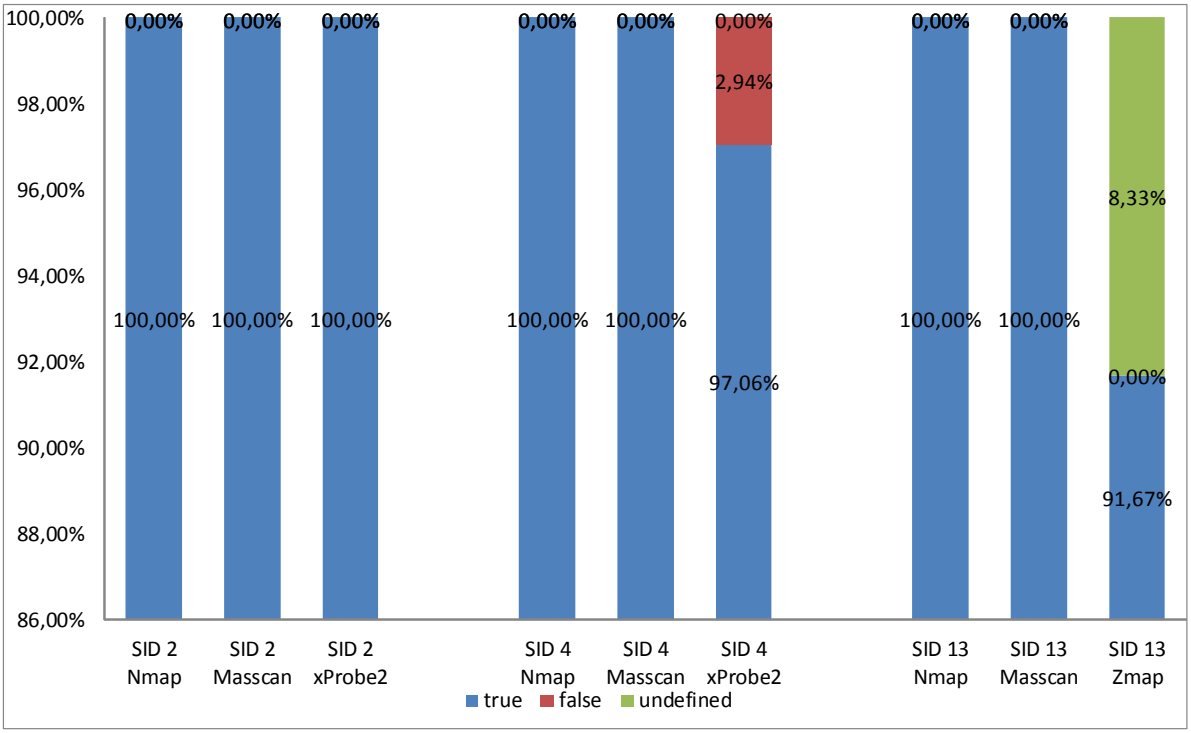


Figure 6.1: The tools' effectiveness in case of SID 2,4 and 13

NMap—*NMap* is able to run the highest amount of different scan methods. Therefore, more scans have been performed with *NMap* compared to the other port scanners. During all scans, *NMap*s results have been very effective. Especially *NMap*s effectiveness in case of vertical TCP scans has been on very high level of effectiveness. Similar to the

other tools, *NMap* sometimes was not able to reach all the hosts targeted. Therefore, some undefined results appeared, too.

In case of UDP scans, *NMap* did not perform with a as high effectiveness as in case of TCP scans, but compared to *xProbe2* and *ZMap*, *NMap*'s results were way more effective.

Masscan—*Masscan* is the most effective tool evaluated in this thesis. Every single port status determined in every single run performed with *Masscan* has been true. Therefore *Masscan*'s effectiveness is the highest possible.

In case of the vertical scans, *Masscan* determined every open port. *Masscan* does not mention closed ports, but it is assumed that ports that are not listed in *Masscan*'s output were determined as closed. This behavior has been verified by analyzing sent and received packets during a *Masscan* scan.

The horizontal and block scans caused a lower effectiveness in case of *NMap* and *ZMap*, because the tools were not able to reach all hosts. In case of *Masscan* every host was reached and scanned successful.

xProbe2—*xProbe2* was analyzed for its effectiveness towards vertical and block scans. To enable block scans with *xProbe2*, it was necessary to iterate scan commands for the single hosts, because *xProbe2* by default is not able to scan several hosts. In case of the horizontal scans *xProbe2* was excluded, because the focus of horizontal scans were on the tools' ability to reach every host with one scan. This aspect can not be compared with one tool iterating the different scans.

xProbe2 determined the ports' status of vertical scanned hosts with an effectiveness a little lower than *NMap*'s or *Masscan*'s effectiveness. It was striking, that *xProbe2* in every scan determined at least one port as filtered. The port number of the filtered port was not fixed. Why *xProbe2* determines at least one filtered port is unknown and implies a lower effectiveness.

The block scans' results, determined by *xProbe2*, were comparable to the tool's results of vertical scans. Every host was reached reliable, but the tool still determined at least one filtered port per host.

ZMap—*ZMap* was only used to perform horizontal scans, because *ZMap* does not offer scans of several ports or port ranges.In case of the horizontal scans, *ZMap* was the tool with the highest amount of undefined results. This was caused by the highest amount of unreachable hosts of all tools.

For example during the performed horizontal TCP SYN scan (SID 14), *ZMap* was only able to scan 22 out of 24 hosts in every run. The fact that the 2 unreachable hosts differed

by each run and that scans with other tools were able to reach at least all hosts one time, implies a low effectiveness. *ZMap*'s is not a very ineffective tool, but compared to the others it is not able to perform the same level of effectiveness.

6.1.3 Efficiency (S.III)

6.1.3.1 Time efficiency

The time efficiency of a port scanner directly influence the tools' quality. Tools with a low time efficiency need more time to perform a certain scan than the other tools.

Setting—The results the time efficiency analysis is based one were determined by measuring the tools needed time during the execution of the scan described in table 5.1. The time was measured from the moment the scan command has been executed until the scanner printed his output and the port scanners activity was finished. This implies, that tools needing more time to calculate their results or to prepare the output have an advantage against tools, that are able to finish calculation faster. Therefore the scan engines speed is not the only variable in this kind of analysis.

It has to be mentioned, that blocks cans were excluded from the analysis of time efficiency, because the results would be dependent on the tools' time efficiency of vertical scans. Because the time needed to scan a host vertical is much more significant than the time needed to start scanning another host.

Result summary—Contrary to the results of the port scanners' effectiveness, the tools' time efficiency differed more striking. Not only the tools, but the scan methods differ a lot, too. Figure 6.2 shows the time needed by different scans performed with *NMap* using different scan methods. On the left, vertical scans of 65535 ports , on the right, horizontal scans of 24 hosts ahve been measured.

The figure shows how big the differences between the scan methods can be. For example a vertical *NMap* TCP Connect scan required 30,97 seconds in average. A scan using the UDP scan methods of the same port amount needed 67.510 seconds (18 hours) in average. This means, that the scan methods' time efficiency differ by the factor of 2000.

The analysis of the tools' time efficiency to each other, resulted in the tools differing significant, too. In case of the vertical scans, *Masscan* by far is the fastest tool. the second fastest tool is *NMap*, followed by *xProbe2*. In case of the TCP SYN scan *Masscan* was 6 times faster than *NMap*.

The most time efficient tool to perform horizontal scans is *ZMap*, directly followed by *Masscan*. The tool with the lowest time efficiency, in case of horizontal scans, is *NMap*.

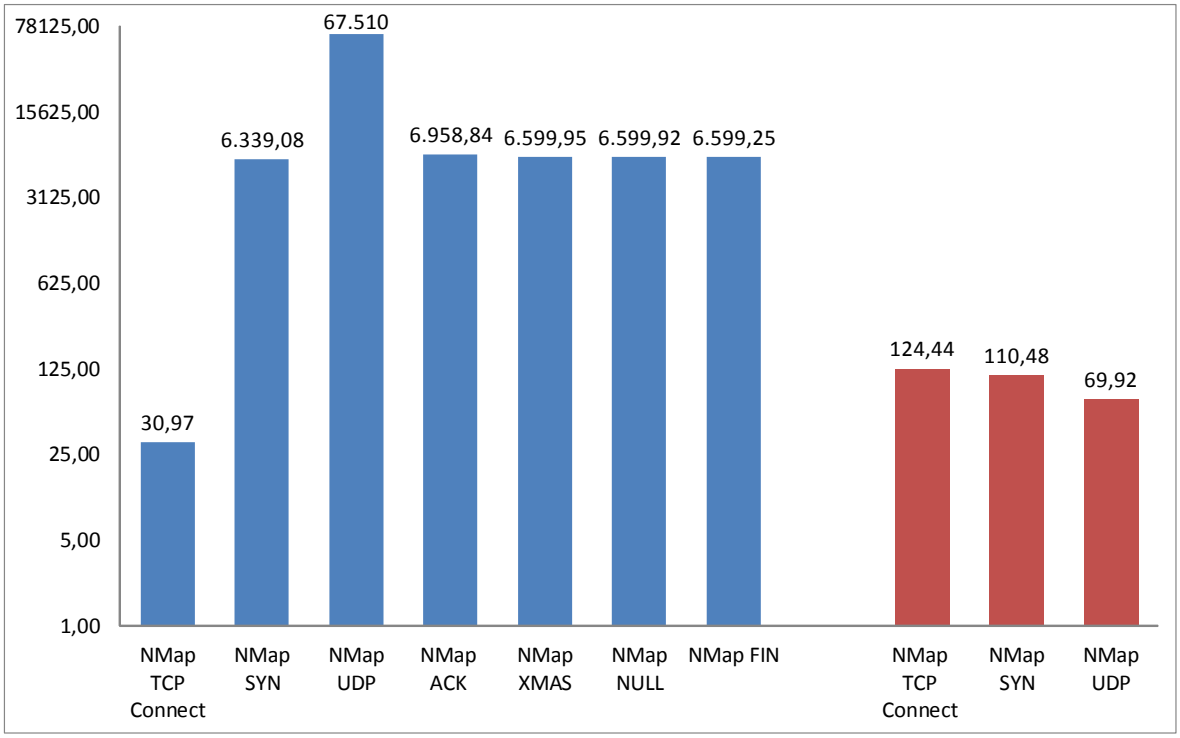


Figure 6.2: Comparison of the time efficiency of the different *NMap* scan methods. On the left hand vertical scans of 65535, on the right hand horizontal scans of 24 hosts. All values in seconds.

For example, *ZMap* needed 8 seconds to scan 24 hosts with a TCP SYN scan on one host. *Masscan* needed 11,33 seconds and *NMap* 110,48 seconds in average.

NMap—*NMap*'s purpose are reliable results and additional benefit thanks to a wide range of capabilities and options. Therefore *NMap*'s time efficiency is not the best. One reason may is the need of time to calculate all outputted results, beside the port scan.

For vertical scans, *NMap* is the second most time efficient port scanner, that has been evaluated. The tool was more efficient than *xProbe2*, but much less efficient than *Masscan*. It is striking, that the different runs of each scan were similar. Especially in case of the NULL, XMAS and FIN scan, the different runs nearly needed the same time. Even the duration of NULL, XMAS and FIN scans are quite equal, as can be seen in figure 6.2.

Compared to *ZMap*'s and *Masscan*'s high time efficiency by performing horizontal scans, *NMap* is muss less efficient. While *Masscan* and *ZMap* needed about 10 seconds to scan 24 hosts using the TCP SYN scan, *NMap* was in need of about 110 seconds in average.

Masscan—Next to its nearly perfect effectiveness, *Masscan* persuades with its very time efficient scans. For both, the vertical and the horizontal scans, *Masscan*'s results were the most efficient, respectively not far from the most efficient results.

The reasons for *Masscan*s ability of time efficient scanning is the tool's purpose, which are fast and large scans.

xProbe2—*xProbe2* is a fingerprinting tool, running several modules to leak as much as possible information from the attacked hosts. The calculation and handling of many different measured values and complicated processes to identify information about the target host may need a lot of time.

This can be observed by *xProbe2*'s time efficiency for vertical scans. *xProbe2* was less time efficient than *NMap*, and therefore the tool wit the lowest time efficiency evaluated. An example of *xProbe2*'s improvable time efficiency was the vertical scan of the 47 selected ports using the TCP SYN scan module. Wile the scan performed with *Masscan* needed 12,33 seconds, and the scans with *NMap* 29,55 seconds in average, *xProbe2* was in need of 5.601 seconds.

ZMap—*ZMap*'s purpose are fast, horizontal scans. This purpose is well-implemented. *ZMap*'s time efficiency in case of horizontals scans were the most time efficient of all tools evaluated. The tools' results only were a little more time efficient than *Masscan*s.

6.1.3.2 Data efficiency

One possibility to compare the conspicuousness of different scans are the needed packets or bytes. Both metrics are considered in the section of the port scanners data efficiency.

Setting—To analyze a port scanners data efficiency, it is necessary to consider the tools' need of packets and bytes. To get significant and comparable results, TCP SYN scans were performed with *NMap*, *Masscan* and *xProbe2*, targeting one host on 10.000 ports. To compare the different scan methods, too, a TCP Connect scan of 10.000 port has been performed with *NMap*. The network traffic during the scans has been captured and analyzed after a scan has been finished.

To separate the traffic caused by the port scan, the network traffic was filtered for the conversation between the attacker and the scanned host. This approach enables to differ packet and byte corresponding to their source. This means one the one hand the total need ob packets and bytes can be evaluated, on the other hand the proportion of sent and received packets or bytes of each port scanner can be analyzed.

Result summary—Regarding figure 6.1.3.2, it can be seen, that the most packet efficient tool evaluated is *xProbe2*, limited to the analyzed scan methods, followed by *NMap*'s TCP Connect scan, *NMap*'SYN scan and *Masscan*'s SYN scan. Consider that packets marked as A⟶V, are packets sent by the attacker, and packets marked as V⟶A, are packets received by the attacker and sent by the scanned host.

This observation includes two surprising aspects. First, *xProbe2* leaks the most information of all tools by running several fingerprinting modules. To do this, the tool does not need significant more TCP packets, than scanned ports in each direction. Second, the TCP Connect scan, which performs the whole *TCP Three Way Handshake*, needs less packets that the TCP SYN scan. The tool with the highest need of packets is *Masscan*.

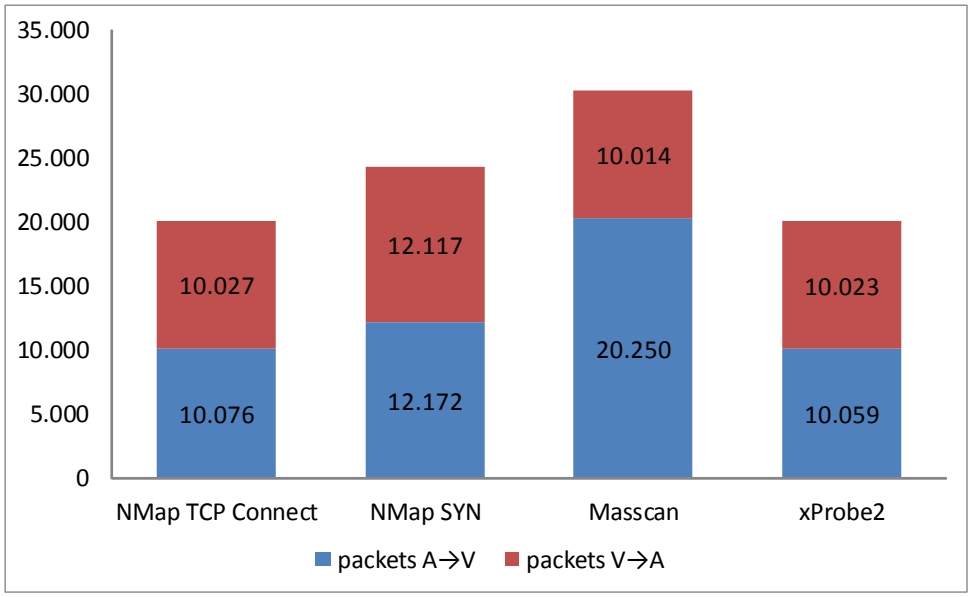


Figure 6.3: Overview of sent and received packets during a TCP SYN scan of 10.000 ports performed with *NMap*, *Masscan* and *xProbe2* and during a TCP Connect scan of 10.000 ports with *NMap*

In case of the tools' byte efficiency, a direct dependency between the need of packets and bytes can be discovered for the most scans and tools. An interesting fact is hidden behind the need of bytes of *NMap*'s TCP Connect scan. While the TCP Connect scans sends less packets to its victims as *NMap*'s TCP SYN scan, but the TCP Connect scan sends more bytes.

NMap—Behind *xProbe2*, *NMap* was the second most data efficient tool evaluated. The fact that 10.076 packets sent during a TCP Connect scan include more byte than 12.172

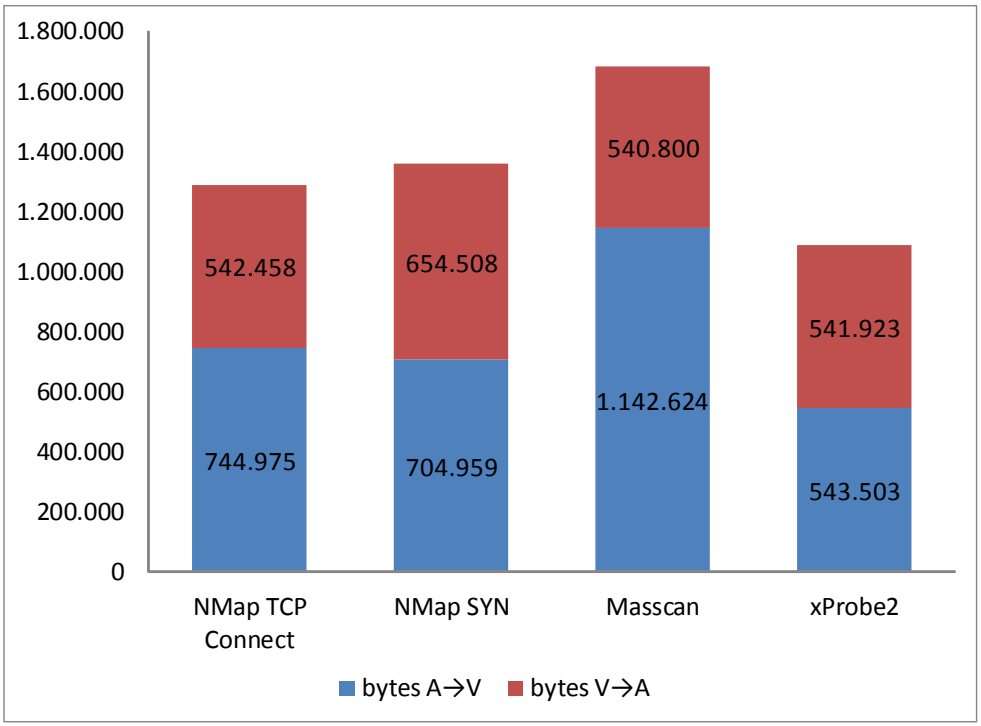


Figure 6.4: Overview of sent and received bytes during a TCP SYN scan of 10.000 ports performed with *NMap*, *Masscan* and *xProbe2* and during a TCP Connect scan of 10.000 ports with *NMap*.

sent packets during a TCP SYNs can was surprising.

Contrary to *NMap*'s improvable time efficiency, the tool's data efficiency is more persuading. At the same time this shows, that the time and data efficiency are to different criteria of analyzing a port scanner.

Masscan—*Masscan* is the most time efficient port scanner for vertical scans, thanks to its high rate of packets per seconds and it optimized scan engine purposed to perform fast scans. But the tool's data efficiency was the lowest one. *Masscan* is in need of more packets and bytes than any other tool evaluated.

The reason for *Masscan*'s low data efficiency are 10.000 TCP Reset packets sent after each run. For each scanned port, *Masscan* creates one packet to reset the started TCP connections during the performed SYN scan. Why *Masscan* behaves this way is unknown. This makes it much more remarkable, that *Masscan* is still faster than *NMap* and xProbe and nearly as fast as *ZMap* in case of horizontal scans.

xProbe2—As already mentioned above, *xProbe2* needs the lowest amount of packets and bytes of all tools. For a scan of 10.000 ports with the TCP SYN scan, the tool only sent 10.059 packets and received 10.023 packets in average.

6.1.4 Randomization S.(IV)

As described in section 2.1 the evaluation of port scanners randomizations needs different points of view. There are different levels of randomization, which are:

- **port order randomization** in case of vertical scans
- **host order randomization** in case of horizontal and block scans
- **host-subnet randomization** in case of scans of several subnets

Setting—The ability to randomize on different levels is analyzed based on results of different kinds of scans. To analyze the port order randomization SYN scans were performed targeting 10.000 ports, beginning with port number one up to port number 10.000, on one host. The traffic of the scans was captured and filtered for the packets sent by the attacks. Afterwards the order of contacted ports has been extracted.

In case of the host order randomization, 24 hosts have been specified with their IP address, and a horizontal scan has been performed. The traffic was captured and filtered, comparable to the process used to analyze the port order randomization.

The host-subnet randomization has been analyzed by specifying two whole subnets instead of the hosts directly. The targets were distributed on both subnets. Through the analysis of captured traffic, the order of hosts could be extracted and mapped on the different subnets.

Result summary—Tablke 6.2 shows the tools' ability to randomize on the three specified levels. The only tool able to randomize every level of port and host order is *Masscan*. *NMap* is able to randomize the port order of vertical scans and the order of several specified hosts.

xProbe2 and *ZMap* are limited in the possibilities of targeting several hosts, in case of *xProbe2*, and to scan several ports with one command, in case of *ZMap*. In their scope both tools are able to randomize the certain levels. *xProbe2* is able to randomize a scan's port order, while *ZMap* scans hosts and hosts from different subnets in a randomized order.

	Port order	Host order	Host-subnet
NMap	✓	✓	✗
Masscan	✓	✓	✓
xProbe2	✓	✗	✗
ZMap	✗	✓	✓

Table 6.2: The tools' abilities to randomize the port order, host order and the host order of several subnets

Port order randomization—The port order randomization, offered by *NMap*, *Masscan* and *xProbe2*, is set as default to all tools. While *NMap* offers the possibility to scan hosts in ascending order, *Masscan* and *xProbe2* always randomize the specified port order.

Host order randomization—The host order randomization allows attackers to scan several hosts by minimizing their risk of being detected and without loosing time. This option is offered by *NMap*, *Masscan* and *ZMap*. Contrary to *Masscan* and *ZMap*, *NMap* has to be configured to randomize the target host order, by its `-randomize-host` command. In case of *Masscan* and *ZMap* the host randomization is default.

Host-subnet randomization—The host-subnet randomization, which can be used to bypass network-based port scan detection tools, only is offered by *ZMap* and *Masscan*. When whole subnets are specified, the tools have to scan every IP address included in the subnets, without the know which IP address is used and which hosts are up. The consideration of host-subnet randomization is an indicator for the tools' focus on large scans.

6.1.5 Configuration of scans and OS detection capabilities (S.V)

How described in section 2.1.5, port scanners offer different possibilities to design and configure scans individually. This aspect is analyzed based on the tools' offered parameters, which can be printed in the Linux terminal, and the tools' documentation.

This criteria is divided in three parts. First 10 different features a port scanner may offers are listed. The features are selected according to their importance and influence on port scans. Second, the tools' configuration possibilities are analyzed and described in detail. Third, *NMap*'s and *xProbe2*'s capability of operating system detection is evaluated.

Result summary—The selected features a port scanner may offers are the ability

- to discover hosts in a specified network range, without port scanning the hosts. This differs the host discovery from scanning whole subnets.

- to specify the target ports as a range of ports.
- to load and execute scans from saved scan scripts.
- to detect the target's operating system.
- to save the results into XML or Json files, to process them automatically.
- to output the trace route to the scanned hosts.
- to customize the flags in the sent TCP headers.
- to scan a port range not randomized.
- to wait a defined amount of seconds, called send delay, between different scan probes, purposed not to penetrate the scan's target to much.
- to limit or specify the sent packet amount per second.

Table 6.3 presents, which feature is offered by which tool. *NMap* offers all selected features. *ZMap* offers only 2 out of 10 features.

	NMap	*Masscan*	*xProbe2*	*ZMap*
host discovery / ping scan	✓	✗	✗	✗
port ranges	✓	✓	✓	✗
script scan	✓	✗	✗	✗
OS detection	✓	✗	✓	✗
XML/Json output	✓	✓	✓	✓
trace route	✓	✗	✓	✗
customize TCP scan flags	✓	✗	✗	✗
consecutively port order	✓	✗	✗	✗
scan delay	✓	✗	✓	✗
packets per second limitation	✓	✓	✗	✓

Table 6.3: Availability of selected port scan capabilities

NMap—*NMap* [15] is known for its wide range of offered capabilities of scan configuration. *NMap* offers over 80 different options, while most of them can be specified with entering a parameter value. This amount is huge considering, that *NMap*'s purpose is to scan ports. The options are divided in different groups. Options within a group influence the tool's behavior in a similar way. The groups and some examples of scan configuration capabilities are described in the following. *NMap* offers different ways to specify a scan's targets. The target specification is able to read the targets from a specified file or even to scan a specified number of unknown randomized hosts. The next group of capabilities are the offered scan techniques. *NMap* offers all Scan techniques considered in this thesis and additional scan like the FTP Bounce scan. An example,

which may conveys *NMap*'s range of possibilities, is the users capability to set every single flag in the used protocol's header.

Next to the choice of hosts and the used scan technique, *NMap* allows different ways to specify the target ports and the way they are scanned. For example *NMap* includes a mode which scans ports more faster, an option to not randomize the port order, the port order is randomized by default, or the possibility to scan the most common scanned ports.

Furthermore, *NMap* can be configured to identify running services on the target hosts and to read scan commands from a specified file. The scan commands', which are read from a file, arguments can be specified in the terminal. Another option included in the capabilities of script scanning, is the output of every single sent and received data packet.

Even additional functions of *NMap* can be configured, such like *NMap*'s OS detection capabilities, which can be configured to guess operating systems more aggressive.

Next to the named examples of *NMap*'s capabilities, aspects influencing *NMap*'s timing and performance, its firewall and ids evasion ability and the tool's output can be configured. Last but not least *NMap* offers a capability group called *Misc* including the possibility to use the IPv6 protocol.

Summarized, *NMap* offers an huge amount of possibilities to configure scans according to the user's needs. Some options are more easy to understand and to use, than others, because some options need knowledge in the field of networks and port scans. Thanks to *NMap*s well-written documentation [27] of all options, which in addition includes the explanation of port scanning basics, the wide range of capabilities is easy to use after a short time of informing.

Masscan—Contrary to *NMap*, *Masscan*'s purpose is not to offer a port scanner with a lot of additional functions and a user-friendly interface, but the ability of fast and big port scans. This fact is reflected in *Masscan*s offer of capabilities to configure scans. *Masscan*'s options can be printed into the terminal by only writing the *Masscan* command without any other parameters. Additional, Robert Graham explains some of the main functions on the tools' git-hub repository website [17].

The capabilities which can be used are a subset of about 30 functions *NMap*s offers. The options are added to a scan command the same way, they are added to *NMap*'s scan commands. The focus of *Masscan*'s capabilities are the fields of output, target specification, firewall evasion and performance. The tool is reduced to the most needed capabilities, to enable fast and efficient scans of big networks. *Masscan* offers the possibility to safe scan results into files and to print saved results from file.

xProbe2—The third tool analyzed for its capabilities of scan configuration is *xProbe2*. *xProbe2* offers about 20 options, which can be selected and parametrized. In difference to the other port scanners evaluated, *xProbe2* does not offer capabilities to improve the port scans' performance or results, but to configure the numerous fingerprinting modules offered by *xProbe2*. In fact *xProbe2*'s port scan module is an additional capability of a fingerprint tool, only, and not the other way around. Therefore port scans performed with *xProbe2*, can not be configured with additional options or parameters.

ZMap—Last but not least *ZMap* is analyzed for its offering of options to configure scans. *ZMap*s purpose are horizontal scans of whole subnets. Therefore the capabilities offered by *ZMap* are focused on the tool's performance, scans, network options, few advanced options and additional options. The basic arguments, which can be set are the target port, a specified output file and a blacklist including subnets, which never should be scanned, according to *ZMap*'s documentation [19].

The tool's scans can be configured in the maximum and minimum amount of scanned hosts, the maximum runtime of the scan and the packet-per-second rate. Additional the seed value can be specified. The calculation of the host permutation is based on this specified seed value.

Furthermore, the source port *ZMap* should use can be set, the source IP address and the used interface can be specified. Last but not least, *ZMap* enable the user to save configurations into a configuration file and log results and events into log-files.

OS detection capabilities—One kind of capability analyzed more detailed is the capability of OS detection. The tools offering the option of operating system identification are *NMap* and *xProbe2*. To analyze the tools quality of OS detection, three runs are performed with each tool targeting a virtual machine running Linux Ubuntu 14.10. The used kernel version was *Linux version 3.16.0*. The scans performed were TCP SYN scans targeting 1000 ports on one host.

Figure 6.1.5 presents *NMap*'s results of the performed OS detection. *NMap*'s results were more constant than xProbe2's. While *NMap* determined the same 3 different operating systems in the different runs, *xProbe2* determined different operating systems in each run. The results of both tools can be seen in table 8.13 in the annex.

6.2 Evaluation of port scan detection tools

This section includes the evaluation of all port scan detection tools presented in section 4.2, considering all criteria described in section 2.2.

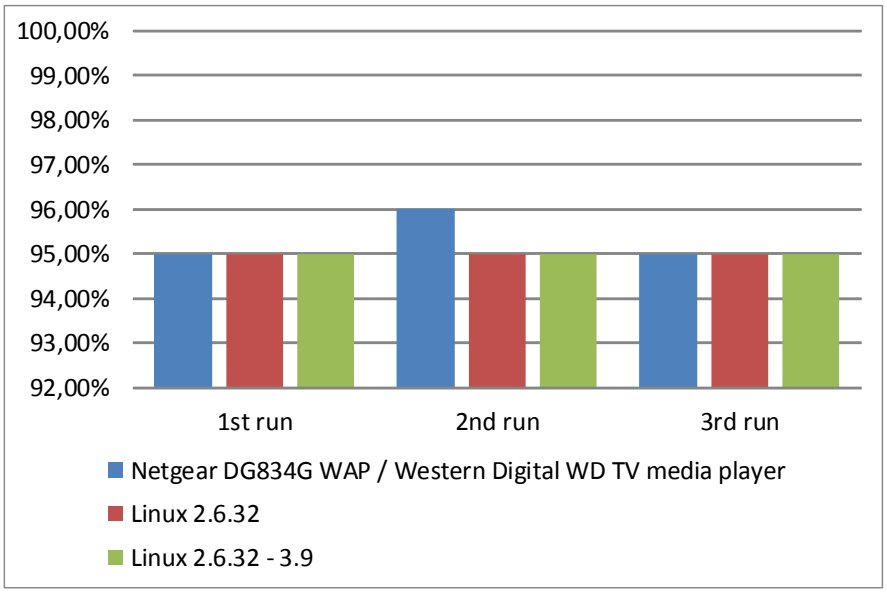


Figure 6.5: *NMap*'s results of the performed OS detection scans

6.2.1 Reliability (D.I)

Setting—The analysis of the port scan detection tools' reliability is based on the port scan alerts written by the port scan detection tools, while different vertical scans of 100 ports have been performed.

The different runs of port scans were distributed over time, to ensure that two scans are not determined as one port scan. The detection tools' alerts are analyzed for the amount of alerted ports. An alerted port is defined as a scanned port, which was detected as being port scanned and the port's number was written in an alert message. Therefore the ratio of alerted ports can be used to compare the tools reliability of detecting certain incoming scans.

The port scans were executed on different attacker VMs, because the attacking hosts were not proposed to be blocked by the port scan detection tools. The attacked virtual machine changed, too, therefore the tools' evaluation is based on the results of different instances of one tool.

One of this analysis problem is to determine how many port have been detected by the tools, because not all tools alert every scanned port. For example *Scanlogd* only mentions 6 scanned ports per scan. In case more than 6 ports have been detected as scanned, the tool shortens the list with 3 dots. This makes it impossible to specify the

amount of detected scanned ports. Therefore, only the amount of alerted ports can be analyzed.

A detailed overview of the port scan detection tools' results can be found in the appendix in the table 8.14 up to 8.17.

Detected scan methods—During the execution of port scans to trigger the detection tools' alerts, it was remarkable, that the tools differ a lot in the scan methods they are able to detect and to alert. Table 6.4 shows which tool was able to detect a certain scan method, performed with a certain tool.

PSAD is able to detect all used scans method and tools, except *NMap*'s ACK scan. The tool, which detects least scan methods and tools is *Portsentry*. THe tool only detected *NMap*'s TCP Connect and UDP scan, and *xProbe2*'s UDP scan.

Scan	*PSAD*	*Portsentry*	*Scanlogd*	*Snort*
NMap TCP Connect	✓	✓	✓	✗
NMap SYN	✓	✗	✓	✓
NMap UDP	✓	✓	✗	✓
NMap ACK	✗	✗	✗	✗
NMap FIN	✓	✗	✓	✓
NMap XMAS	✓	✗	✓	✓
NMap NULL	✓	✗	✓	✗
xProbe2 TCP	✓	✗	✓	✓
xProbe2 UDP	✓	✓	✗	✓
Masscan	✓	✗	✓	✓
method detection ratio	0.9	0.3	0.7	0.7

Table 6.4: Overview of the detection tools ability to detect certain scan methods

Result summary—The port scan detection tools differed a lot in their ratio of alerted ports. Table 6.5 presents how many ports of 100 scanned ports were mentioned by each tool in average.

The most reliable tool evaluated is *PSAD*. *PSAD* detected and alerted 86% of all scanned port in average. This is by far the highest ratio of alerted ports. The less reliable tool is *Portsentry*. *Portsentry* only alerted 2% of all scanned ports. *Scanlogd* may detected more ports as it wrote into its alerts, but only the amount of named ports can be used to evaluate the tool. The second most reliable tool is *PSAD*, with a ratio of alerted ports of about 59%.

Portsentry—*Portsentry* is the tool, with the lowest reliability. Only 3 scan methods have been detected and only few ports have been alerted. The reason for *Portsentry*'s bas

Scan	*PSAD*	*Portsentry*	*Scanlogd*	*Snort*
NMap TCP Connect	100	8	> 6	0
NMap SYN	100	0	> 6	73.33
NMap UDP	94.66	6.33	0	91
NMap ACK	0	0	0	0
NMap FIN	99.66	0	> 6	100
NMap XMAS	99.66	0	> 6	100
NMap NULL	99.66	0	> 6	0
xProbe2 TCP	100	0	> 6	75.66
xProbe2 UDP	66.66	6	0	72.66
Masscan TCP	100	0	> 6	76.66
RATIO OF ALERTED PORTS in %	86.03	2.03	> 4.2	58.93

Table 6.5: Overview of all determined average ratios of alerted scanned ports for three runs of scans from port one up to port 100

results my is the tool's default configuration and its approach to only alert ports, that are open.

PSAD—THe most reliable tool evaluated is *PSAD*. *PSAD* detected and alerted the whole scanned port range in case of TCP Connect and TCP SYN scans. In case of the FIN, XMAS and NULL scan performed with *NMap*, the tool was not able to detect one scanned port in all 3 runs. This implies an average of alerted ports of 99.66%.

In case of the UDP scans performed with *NMap* and *xProbe2*, *PSAD*'s results were not as good as the results of the other scan methods, but still more reliable than Portsentry's and Scanlod's results.

Figure 6.2.1 shows the average of *PSAD*'s and *PSAD*'s ratio of alerted ports for each performed scan method. *PSAD* is the second most reliable tool, but was not as able as *PSAD*, to determine the scanned port range.

Scanlogd—As mentioned, *Scanlogd* does only write up to 6 port numbers in its alert messages. The tool detected 7 scan methods and alerted 6 port and more for each scan. The exact amounts of alerted ports are unknown.

Therefore, *Scanlogd*'s reliability can not be evaluated as detailed as the other detection tools, because of a lack of information. It is assumed, that the tool detected more ports as alerted, but there is nor prove for *Scanlogd*'s higher reliability.

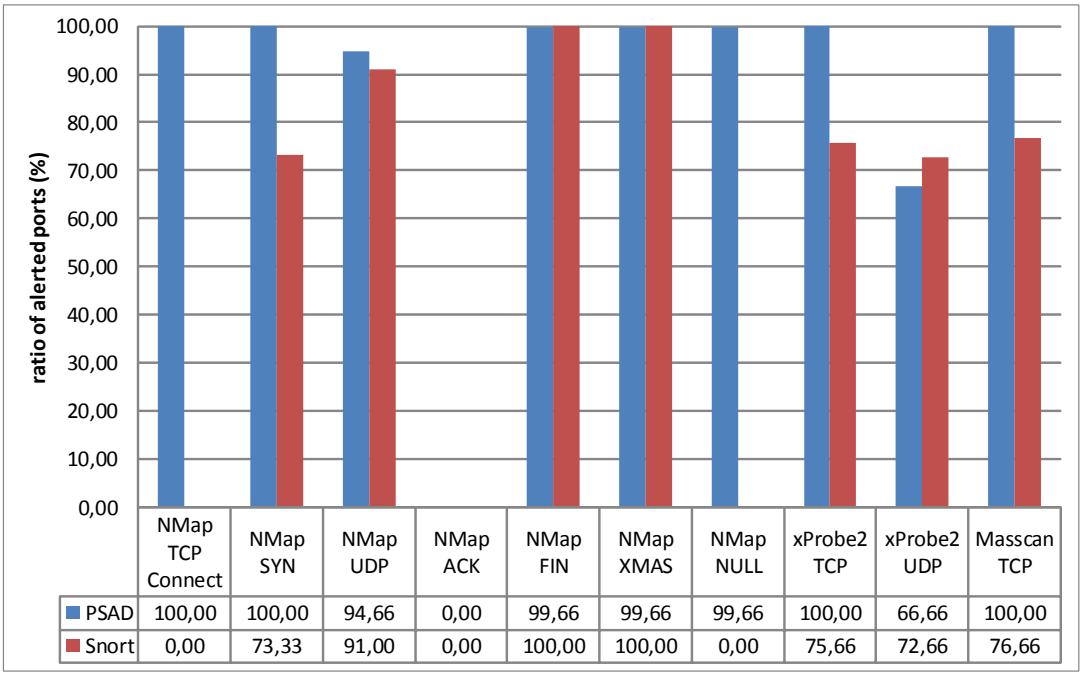
Snort—The network intrusion detection system *PSAD* was not able to detect *NMap*'s TCP COnnect, UDP and NULL scan. Contrary to *PSAD*, *PSAD* was not able to specify

the scanned port range as good as *PSAD* does. Therefore *PSAD*'s reliability is the second best only.

NMap's FIN and XMAS scans were not alerted as port scan attacks by *PSAD*. The tool did not write any port scan alerts in the log-file, but determined the FIN and XMAS scans as try of information leak, therefore the alerts were written to another log-file.

In this case, *PSAD* benefits of its additional features as a complete intrusion detection system. Even if the XMAS and FIn scans not have been identified as port scans, the user is informed about an incoming attacks on certain ports.

Figure 6.2.1 shows *PSAD*'s and *PSAD*'s results of the analysis of ratio of alerted ports.



	NMap TCP Connect	NMap SYN	NMap UDP	NMap ACK	NMap FIN	NMap XMAS	NMap NULL	xProbe2 TCP	xProbe2 UDP	Masscan TCP
PSAD	100,00	100,00	94,66	0,00	99,66	99,66	99,66	100,00	66,66	100,00
Snort	0,00	73,33	91,00	0,00	100,00	100,00	0,00	75,66	72,66	76,66

Figure 6.6: Ratio of alerted ports of several scans detected by *PSAD* and *PSAD*

6.2.2 Anti-scan reaction (D.II)

This aspect of evaluation summarizes the port scan detection tools ability of reacting after a port scan attack has been detected. The possible reactions are separated in passive and active reactions. Passive reactions do not directly influence the attacker's results, active reaction do directly influence them. First all reactions offered at least by one of the evaluated tools are explained. Table 6.2.2 presents which tool is able to

perform which anti-scan reaction.

Passive reactions—The most simple and most common reaction after a port scan has been detected, is to write an alert message to a certain log-file. This reaction is provided by all tools, because it is the most simple way to inform the user about an incoming attack.

Another reaction which is purposed to inform the user about an attack is to write e-mails including alert messages. This reaction ensures, that the detection tools administrator can react immediately. These two reactions does not influence the attackers results directly.

Active reactions—In difference to the reactions purposed to inform the user, blocking directly influences the port scans results. Blocking means to use the host's firewall, to prevent any conversation between the attacker and the target. This means the port scanner will not receive any information, after the attacker's IP address have been blocked, because it looks like the scan's target host is down.

Another reaction against port scan attacks is to redirect incoming scans. If the scan's source is identified, the traffic received from this source can be redirected to another host, which can be a dead - or dummy - host. This reaction is effective, because, in difference to blocking the attacker's IP address, the attacker does not know, that his scan has been redirected and detected.

A very special reaction after an port scan has been is detected is a port banner message, shown if the port scan detection tool detects the scan. This measurement, only provided by *Portsentry*, enables the user to print a text message into the attackers results. The port banner measurement does not influence the port scanner's results.

Last but not least, *Portsentry* offers the possibility to start external command after a scan has been detected. This is not a direct reaction towards port scan attacks, but enables the user to start external commands, reacting on the detected scan.

Table 6.2.2 shows, which reaction is offered by each port scan detection tool.

tool	alert	e-mail	blocking	redirection	port banner	external command
Portsentry	✓	✗	✓	✓	✓	✓
PSAD	✓	✓	✓	✓	✗	✗
Scanlogd	✓	✗	✗	✗	✗	✗
Snort	✓	✗	✗	✗	✗	✗

Table 6.6: Overview of offered anti-scan reactions

6.2.3 Usability (D.III)

Contrary to the port scanners, the port scan detection tools may are more difficult to set up and integrate. This is caused by the need of certain firewall settings and communication chains between the firewall and the port scan detection tool. Therefore it is a criteria of quality, which steps are necessary to integrate the port scan detection tool to a system and how much knowledge about networks and firewalls are necessary. This aspect's evaluation is based on observations made during the integration of the tool to the chair for network architectures and services at the Technical University Munich.

Following characteristics of the port scan detection tools are described:

- How the tools can be installed
- the steps necessary to integrate the tools
- different modes to use offered by the tools
- the tools' documentation

Portsentry—*Portsentry* can be installed by using Ubuntu's `apt-get` command or comparable commands on other Linux versions. *Portsentry* was able to communicate with the used firewall, which was IPtables, from the beginning and was able to detect scans without ny configuration. The tool was able to restart after every reboot without any errors. The first configuration of *Portsentry* is simplified by a well documented configuration file and a helpful help - command, which prints the tools possibilities of running *Portsentry*. Thanks to the well documented configuration file, nearly none knowledge about the fields of network, port scans of firewalls is needed.

Portsentry can run six different modes of port scan detection. The available modes are: the `tcp` mode, which implicates default port scan detection methods. To detect stealthy scans, too, *Portsentry* offers the `stcp` mode or `Stealthy TCP` mode. The third mode to detect TCP scans is the `advanced TCP` or `atcp` mode covering a wirde range of different methods. The advantage of the `atcp` and `stcp`, according to the comments in the configuration file **??**, is a higher risk of alerted false - positives, scan alerts without an incoming attack. According to the three TCP modes, *Portsentry* offers the same modes for the UDP protocol.

PSAD—Comparable to *Portsentry*s, *PSAD* can be installed using Ubuntu's `apt-get` function or comparable Linux commands. After the installation has started, the user is asked to enter some information, which are directly written into *PSAD*'s configuration file. The required information is for example the e-mail address to write e-mail alerts and the wished host name.

In difference to *Portsentry*, *PSAD* required more specific firewall settings before it is able to detect port scan attacks. Sadly there are no official advices how to configure the firewall and unofficial sources like blogs or forum entries do not always help or give false instructions. After the right settings have been done, *PSAD* started alerting scans. It was not necessary to adjust the settings of the firewall or *PSAD* after the right settings have been found.

The first configuration of *PSAD* may is more time expensive than the configuration of *Portsentry*. This is caused by a lot of possibilities to modify the configuration. Thanks to a well-commented configuration file, The configuration can be done easily, but, as mentioned, needs more time.

As analyzed in section 6.2.4 *PSAD* offers an overview of detected port scans in addition to common alerts. This makes *PSAD* more friendly to the user. Summed up, *PSAD* requires a bit more knowledge than *Portsentry*, but after its setup and integration it is as easy to use as *Portsentry*.

Scanlogd—In difference to the other host-based port scan detection tools, *Scanlogd* has to be installed from its binaries. After *Scanlogd*'s installation nothing has to be configured or modified. Right after the installation *Scanlogd* was able to detect the first scans and wrote the alerts to syslog by default. The user is not even informed about the tools correct integration or its status.

Scanlogd can be treated as Linux service, and can be started, stopped or restarted via the Linux terminal. As already observed in other aspects of comparison, *Scanlogd* is a simple to integrate but not suitable to use tool. The tools' output and configuration possibilities are limited to a minimum. A not-well commented documentation and a short `readme` -file do not help to familiar with the tool fast.

Snort—While the host-based port scan detection tools has to be installed on the host which should be monitored, *Snort* and its port scan detection module SFPortscan can be installed on any host in a network and is able to monitor the whole network traffic with a monitoring port. Therefore a *Snort* installation has to be done and a monitoring port is required to monitor several hosts with the SFPortscan module. The integration of *Snort* and its monitoring port may require more knowledge of the field, than the host-based detection tools require.

After *Snort* has been integrated and it is able monitor the specified hosts, SFPortscan has not to be installed, because it is already part of *Snort* and only needs to be configured in *Snort*'s configuration file, as described in section 6.2.5.

Therefore SFPortscan itself is a very comfortable, easy to use way to detect port scans. Nevertheless the integration of *Snort* and the required network monitoring are more

	Portsentry	*PSAD*	*Scanlogd*	*Snort*
date & time	✓	✓	✓	✓
scanned port range	✗	✓	✗	✓
alert for every single port	✓	✗	✗	✗
source IP address	✓	✓	✓	✓
scanned host	✓	✓	✓	✓
protocol	✓	✓	✗	✓
scan method	✗	✓	✗	✗
alerts to log-file	✓	✓	✓	✓
history file	✓	✓	✗	✓

knowledge and time expensive than the integration of host based detection tools. It has to be mentioned, that SFPortscan's documentation is well-written and includes a section about tuning SFPortscan [26] to achieve better results. This is a very helpful benefit regarding *Snort*'s and SFPortscan's usability.

6.2.4 Output quality and format (D.IV)

This part of the evaluation is based on observations during the analysis of the other aspects of the evaluation. As already mentioned in section 6.2.1 the analysis of reliability, the structure of the outputted alerts differs a lot. This obtains for the used log files and the alerts' quality.

Result summary—Table 6.2.4 shows, which tool offers a certain aspect of output quality and format. The tools were analyzed for the information included in the tools' alerts, their matter of printing alerts and the location of alerts. The matter *Snort* prints its alerts, is the most understandable and readable way the evaluated port scan detection tools use to output their results. The approaches of the other tools similar in their informative content and their structure.

Outputfiles—First the output and log files are analyzed. Incipient with *PSAD*, it is analyzed where the alerts are written. *PSAD* writes alert messages into a specified file, in this example the messages are written to Linux's `syslog` file. Additional *PSAD* offers an overview over the last detected attacks including statistical values and the most active attackers. This information is written into a file called status.out.

Comparable to *PSAD*, *Portsentry* writes alerts into a specified log-file, in this case `syslog`, too. Beside this source, no information about the detectors work or results are available as output.

Third Scnalogd's output files are analyzed. Such as *Portsentry*, *Scanlogd* only writes alert messages into `syslog`. The file *Scanlogd* writes into can not be specified or changed.

Last but not least *Snort* offers the possibility to specify an own port scan attack log-file. In this log-file only port scans are written and therefore it is possible to get a quick overview over incoming or detected port scans.

PSAD's alerts—*PSAD*'s alert messages include the time, the port scan attack has been alerted, the used scan method, the attacker's IP address, the scanned range of ports and the set flags in the analyzed data packets. Especially the output of the used scan method is a very helpful feature *PSAD* offers. It enables to estimate the attacker's purpose and what information the attacker leaked or wanted to leak. Figure **??** shows an example alert message written by *PSAD*. The performed scan was a TCP Connect scan. Therefore *PSAD* was not able to clearly identify if the detected scan is a SYN scan or a TCP Connect scan, because both scans use TCP SYN packets. The attacker and target are clearly noticeable, such as the scanned port range.

```
date&time hostname \textit{PSAD}: scan detected(\textit{
    NMap} -sT or -sS scan):
172.16.0.101->172.16.0.100tcp:[1-100]flags: SYN tcp pkts
    :100 DL:3
```

Listing 6.1: Example of a port scan attack alert written by *PSAD*

But *PSAD* offers another source of information, which can be used to get an quick overview about the main characteristics of the last few detected port scan attacks. *PSAD* offers a command, which prints a file called `status.out` to the terminal. It includes information like the top 50 scan signatures, the top 25 attackers, according to the detected scan size, the top 20 scanned ports, according to the amount of hits, Ip addresses that have been blocked, counters for TCP and UDP scans and other information related to the detected port scan attacks. This overview is very helpful to users, that do not want to open and read the whole log-file, but want to inform about the detection tools status.

Portsentry's alerts—As described above *Portsentry* uses `syslog`, to communicate with the user. The alert messages are similar to *PSAD* one's but differ in the way scanned ports are alerted. *Portsentry* does alert every single scan detected, while *PSAD* combine several probes to one big scan. This causes, that *Portsentry*'s output is not clear, if several port's are scanned and alerted. Next to the number of the scanned port, Portentry's alerts include the time the scan has been detected, the attacker's IP address and the used protocol.Figure 6.2 shows an example alert written by *Portsentry*. The tool detected a TCP scan on port 80.

```
date&time hostname \textit{Portsentry}[1851]: attackalert:
Connect from host: 172.16.0.101/172.16.0.101 to TCP port:
    80
```

Listing 6.2: Example of a port scan attack alert written by *Portsentry*

Scanlogd's alerts—Third the alert messages and structure of *Scanlogd* are evaluated. As already seen during analyzing *PSAD* and *Portsentry*, *Scanlogd* writes its alerts in single lines. First the time and date the scan was detected is outputted, followed by the attacker's and target's IP address and a list of scanned ports. Sadly *Scanlogd* only writes a limited number of ports, because *Scanlogd* does not combine single attacks to bigger scans and only uses one line per alert. *Scanlogd* presents some additional information like the type of service and time-to-live found in the analyzed received packets. Figure 6.3 shows an example for *Scanlogd*'s alert messages.

```
date&time hostname \textit{Scanlogd}: 172.16.0.101 to
    172.16.0.100
ports 21,80,22,23,47,95,..., fSrpauxy, TOS00, TTL 64 @
    time
```

Listing 6.3: Example of a port scan attack alert written by *Scanlogd*

Snort's alerts—Comparable to *PSAD*, *Snort* combine scans detected on single ports to bigger scan attacks. This is a benefit towards clarity in log-files and the ability to get an fast overview of the last few port scan attacks. Next to the mentioned port range, *Snort* writes the date and time, the attacker's and target's IP, a priority value and few more port scan related values into its log file. Contrary to the other tools, *Snort* does print his output in several lines. This enables a more clearly log file structure, a better overview and more informative content. Figure 6.4 shows a typical port scan attack alert, like *Snort* prints it into the specified log file.

```
Time: 06/30-18:31:57.736565
event_ref:0
172.16.0.101->172.16.0.100{portscan}TCP portscan
Priority Count: 5
Connection Count: 10
IP Count: 1
Scanner IP Range: 172.16.0.101:172.16.0.101
Port/Proto Count: 10
```

```
Port/Proto Range: 9:96
```

Listing 6.4: Example of a port scan attack alert written by *Snort*

is not necessary to configure many options.

6.2.5 Configuration of detection (D.V)

This aspect evaluates the tools' possibility to be configured. The tools' configuration directly influence the port scan detection tools' results and the quality of their output. On the one hand tools offering a wide range of configuration possibilities are a benefit for the users which want to individually modify their detection tool, on the other hand too many configuration possibilities may overcharge users, that do not have enough knowledge of the fields of networks and port scan detection.

Portsentry—*Portsentry* uses a configuration file to save and load its configuration. The configurations offered by *Portsentry* inter alia are the ports, that have to be monitored, or certain ports, that should be excluded of *Portsentry*s monitoring. According to a comment in *Portsentry*'s configuration file [21], *Portsentry* should not be configured to monitor more than 1024 ports. The reason for the proposed maximum of monitored ports according to *Portsentry*'s developers is the rising rate of false alarms.

Portsentry uses a `host-ignore -` , a `history -` and a `blocked-hosts -` file, to safe results and note exceptions. These files' paths can be specified in the configuration, too.

It can be configured, how *Portsentry* should react after a scan has been detected. As described in section 6.2.2 *Portsentry* offers a wide range to react against detected scans. The tool can be configured to not do anything, to auto-block the attacker or to perform an external command. Another method that can be configured is to use the route to another hosts, most-suitable a dead one, which can be used to redirect the port scan probes. *Portsentry* enables the user to enter a port banner message into the configuration file. As described in section 6.2.2 can be included to the attackers scan results.

Another value which can be modified is *Portsentry*'s scan trigger. The scan trigger is the amount of scanned ports which triggers a port scan alert. This value directly influences the risk of false alarms on the one, and the chance to detect small scans, too, on the other hand.

In summary, *Portsentry* offers a lot of helpful possibilities to be configured. The focus is set on the tools behavior after a scan detection and less on the parameters of scan detection methods or of the scan detection engine.

PSAD—Comparable to *Portsentry*, *PSAD* uses a configuration file [23] , too. In difference to *Portsentry*, *PSAD*'s configuration file can be located anywhere, as long as the file's path is specified to the tool by entering it in a terminal command.

PSAD first wants the user to specify his e-mail address and the host's name it is running on. After this, *PSAD* can be configured in many different aspects of the running scan detection engine. Inter alia *PSAD* can be configured to use certain IPtable - or other firewall chains, different danger levels of scans can be set, by entering the critical values , which are necessary to reach a certain danger level. Furthermore, the periods of time between two firewall chain checks *PSAD* uses, the scan trigger value, and the alerting methods can be configured. In addition, the log-file for scan alerts can be specified, too.

In difference to *Portsentry*, *PSAD* does not require to mention all ports which have to be monitorred, but enables the user to exclude certain ports. All other ports will be monitored. At all, *PSAD* enables the user to modify the scan detection engine more in detail as *Portsentry* does. Aside from the named configuration possibilities, *PSAD* can be configured to detect scans using the IPv6 protocol, too.

PSAD features to send e-mail alerts to the user's e-mail address. Because a lot of port scans may would lead to a plenty of alert e-mails, the suer can set the maximum amount of sent e-mails and the frequency alert e-mails are sent in. Concluding, *PSAD*'s configuration is very suitable for users, that want to modify their detection tool in detail and not for users, that do not want to invest the required time to configure their detection tool.

Scanlogd—*Scanlogd* does not provide a configuration file, which can be found or edited by the user. There are no sources including information about *Scanlogd*'s configuration or something similar. Therefore it has to be supposed, that *Scanlogd* does not offer any possibilities of configuration. At least, Scanlogd is an easy to use port scan detector, without any features.

Snort—*Snort*'s module SFportscan is evaluated. The way *Snort*'s port scan detection module is configured differs to the configuration files of *PSAD* and *Portsentry*. *Snort* provides one configuration file, where all modules can be activated and configured. SFportscan allows the user to choose from the following options: first the considered protocols can be specified, *Snort* can detect TCP, UDP, ICMP and IP scans. Next to the protocol, different types of scan can be differed by *Snort*. The scan types are described in section 4.2.2. It can be chosen which types of scans should be alerted.

Next, *Snort* offers the possibility to activate three different sense levels, which influence the risk of false alarms. Other possibilities to configure *Snort* is to specify the monitored hosts, to list attackers that will not be alerted and to list hosts which can be scanned without alerts. Last but not least a log-file for port scan alerts only can be specified. In

summary, SFportscan can be configured in the most essential aspects. Thanks to *Snort*s well-implemented scan detection engine, it is not necessary to configure many options.

6.2.6 Approach of port scan detection (D.VI)

In this section the approaches the port scan detection tools use are described and compared.

Portsentry—*Portsentry*, by default , tries to separate normal traffic from scan traffic by ignoring ports a service is bounded to. The user is able to specify the ports to monitor in *Portsentry*'s configuration file. Ports that are listed to monitor are monitored even if services are bound to them. A reason for this behavior could be the avoidance of false-positives, which means the avoidance of alerts without a running scan.

Another method *Portsentry* uses to identify scan traffic is to log all packets with an IP header length less than five. The RFC 791 [5] defines five as the minimum length of IP headers. This method extension seems simple, but may is effective. Port scanners often create data packets themselves to enable fast and efficient scanning. Packet creation engines implemented without care of RFC or other standards may trigger this kind of scan detection approach.

Portsentry uses an own engine to determine previous connections. This is done with an two dimensional array of IP addresses and a counter variable for each port, according to [22]. Always, a hosts connects to a port, *Portsentry* increments the counter in the port's array. If a counter hits a trigger value in a certain time *Portsentry* throws an alert.

PSAD—In difference to *Portsentry*'s port scan detection approach, *PSAD*'s approach is less known. According to an article published by [23], *PSAD* uses firewall messages to determine port scan attacks. Furthermore *PSAD* analyses header flags in case of TCP scans, to guess the scan's scan method or other options available by *NMap*. This enables *PSAD* to detect patterns in detected scans. Last but not least, *PSAD* makes use of TCP, UDP and ICMP signatures, which are part of *Snort*'s intrusion detection system.

Scanlogd—*Scanlogd* expects seven access tries to privileged or 21 hits on non-privileged ports within three seconds to characterize the traffic as a port scan, according to the tool's manpage [28].

Which ports are privileged and which ports are non-privileged is not known, but privileged ports are defined as ports offering an important or well-known service.

It is not necessary, that exactly seven privileged or 21 non-privileged ports are accessed, combinations of them are valid, too. If *Scanlogd* detects five different scans within 20 seconds, it alerts the port scan and stops logging the attack events for few seconds.

Snort—*Snort*'s port scan detection module SFPortscan uses a different approach to detect port scan attacks. SFPortscan looks out for responses sent from closed ports after they have been scanned. In the tool's `readme` -file [29], the developers explain, that "negative responses from hosts are rare, and rarer still are multiple negative responses within a given amount of time." [29]. Therefore the module tries to detect negative responses from closed ports. In addition, SFPortscan is able to determine the different port scan techniques, that can be performed with *NMap*.

Furthermore, *Snort*'s port scan detection preprocessor uses three patterns to separate port scan traffic from other network traffic. First, *Snort* looks out for TCP and UDP traffic, which includes hits on many different ports, between one host and another. Second, *Snort* alerts traffic from many different hosts towards a single target, hitting many different ports. Third, *Snort* is alarmed, if one host talks to many targets on one single port. These patterns match the definition of the different types of *Snort*'s port scan classification.

The approaches of the evaluated port scan detection tools differ a lot. On the one hand, *Scanlogd* and *Portsentry* try to detect port scans by analyzing incoming packets and incorrect packets. On the other hand *Snort*'s module SFPortscan tries to detect floods of negative responses, sent by closed ports after they have been scanned. *PSAD*'s approach keeps quiet unknown, but the tool analyses firewall messages, so it can be assumed, that incoming packets are analyzed, too.

Chapter 7

Conclusion

The evaluation of port scanners and port scan detection tools needs several criteria to compare the tools on different levels. The fact that the products of both kinds of tools can differ a lot, makes it hard to develop metrics which are universal. For example the purposes of port scanners differ in basic aspects such as their ability to scan several ports or hosts with one scan. Tools like *NMap* and *Masscan* regard port scanning from two contrary point of views. *NMap* is implemented to enable stealthy and informative scans, while *Masscan*'s purpose is a high efficiency and mainly fast scans of big networks.

Therefore there is no best port scanner. For users interested in fast, wide-ranged scans, *Masscan* and *ZMap* can be recommended. *NMap* and *xProbe2* are noticeable slower than the other tools, but provide many helpful features to leak more information about the scanned targets.

Comparable to the port scanners, the port scan detection tools have their advantages and disadvantages, too. Tools like *Portsentry* and *Scanlogd* are integrated fast into the used system, but their reliability and output quality is not developed enough. *PSAD* and *Snort*'s port scan detection module SFPortscan are more difficult to integrate, but offer a better reliability and more options to configure. One of *Snort*'s biggest advantages is the tool's ability of detecting horizontal port scans, too. An ability missed by all host-based port scan detection tools, because they only analyze traffic incoming to our out coming from one host.

The test-bed and the set test environment worked well. The criteria can be analyzed based on system-independent results, which reliability is high. Thanks to the integrated controlling network for the running virtual machines, the network traffic related to the tests can be analyzed without influence of the communication with the VMs.

In case of the port scan detection tools, more research will be necessary to improve the tools' quality. One of the main challenges will be the grouping of different detected scan probes and the range of detectable port scan methods. Even the output of the most

tools can be improved, especially in case of *Scanlogd* and *Portsentry*.

This approach of evaluating port scanners and port scan detection tools can be extended. For example distributed port scans can be analyzed for their efficiency and their influence towards the risk of being detected by port scan detection tools. Another extension could be the influence of simulate network traffic, that may causes more false - positives on the detection tools' side.

Last but not least, it is easier to scan a port or to use a port scanner, than detecting port scans. One development observed in the related work and literature, is to define theoretical approaches of port scan detection. In case of SFPortscan, a metric called *closed scan size*, described in Stuart Staniford, James Hoagland and Joseph McAlerny's paper *Practical automated detection of stealthy portscans* [6], is used to differ port scan traffic from normal network traffic. This trend should be followed, to improve port scan detection and enable the as early as possible detection of incoming attacks.

Chapter 8

Appendix

port	service	description
1	tcpmux	TCP port service multiplexer
5	rje	Remote Job Entry
7	Echo	Echo service
19	Chargen	Character Generation service; sends endless stream of characters
20	ftp-data	File Transfer Data
21	ftp	File Transfer Control
22	ssh	SSH Remote Login Protocol
23	telnet	Telnet service
25	smtp	Simpler Mail Transfer Protocol
53	domain	Domain Name Server
80	http	Hypertext Transfer Protocol
102	iso-tsap	X.400, ISO-TSAP Class 0
110	pop3	Post Office Protocol 3
119	nntp	Network News Transfer Protocol
135	loc-srv/epmap	MS DCE RPC end-point mapper
137	netbios-ns	Netbios Name Service
138	netbios-dgm	Netbios Datagram Service
139	netbios-ssn	Netbios Session Service
143	imap4	Interner Message Access Protocol 4
389	ldap	Lightweight Directory Access Protocol
443	ssl	Hypertext Transfer Protocol over TLS/SSL
445	microsoft-ds	Windows 2000/XP SMB
554	rtsp	Real Time Stream Control Protocol
636	sldap	Lightweight Directory Access Protocol over SSL
750	kerberos-iv	Kerberos version 4 (v4) services
765	webster	Network Dictionary
767	phonebook	Network Phonebook
993	imaps	Internet Message Access Protocol 4 over TLS/SSL
995	pop3s	Post Office Protocol 3 over TLS/SSL
1214	kazaa	Kazaa / Morpheus
1433	ms-sql-s	Microsoft SQL Server
1434	ms-sql-m	Microsoft SQL Monitor
1755	ms-streaming	Microsoft Media Player
1863	msnp	Microsoft MSN Messenger
3112	ksysguard	KDE System Guard
3185	smpppd	SuSE Meta PPPD
4000	icq	ICQ/Terabase
4661		eDonkey
4662		eDonkey
4665		eDonkey
5050	mmc	Multimedia Conference Control Tool
5060	sip	SIP
5190	aol	AOL Instant Messenger / Mirabilis ICQ
6346	gnutella-svc	Gnutella
6347	gnutella-rtr	Gnutella
7070	arcp	Realplayer Server
7071	arcp	Realplayer Server

Table 8.1: Overview of ports defined as *selected ports*

Scan	Duration in sec	Open	Closed	Filtered	No Information
NMap SID 1 1st run	31.65	48	65487	0	0
NMap SID 1 2nd run	31.80	48	65487	0	0
NMap SID 1 3rd run	29.48	48	65487	0	0
NMap SID 1 AVG	30.97	48	65487	0	0

Table 8.2: SID 1 results

Scan	Duration in sec	Open	Closed	Filtered	No Information
NMap SID 2 1st run	6427.17	48	65487	0	0
NMap SID 2 2nd run	6340.49	48	65487	0	0
NMap SID 2 3rd run	6249.58	48	65487	0	0
NMap SID 2 AVG	6339.08	48	65487	0	0
Masscan SID 2 1st run	1313	48	65487	0	0
Masscan SID 2 2nd run	1313	48	65487	0	0
Masscan SID 2 3rd run	1312	48	65487	0	0
Masscan SID 2 AVG	1312.66	48	65487	0	0
xProbe2 SID 2 1st run	7848.52	48	65486	1	0
xProbe2 SID 2 2nd run	10898.03	48	65486	1	0
xProbe2 SID 2 3rd run	9206.02	48	65486	1	0
xProbe2 SID 2 AVG	9317.52	48	65486	1	0

Table 8.3: SID 2 results

Scan	Duration in sec	Open	Closed	Filtered	No Information
NMap SID 3 1st run	71463.64	35	65500	0	0
NMap SID 3 2nd run	65535.31	35	65500	0	0
NMap SID 3 3rd run	65533.55	31	65500	4	0
NMap SID 3 AVG	67510.83	33.66	65500	1.33	0
xProbe2 SID 3 1st run	10966.04	33	669	64833	0
xProbe2 SID 3 2nd run	11693.55	33	669	64833	0
xProbe2 SID 3 3rd run	11086.04	32	669	64834	0
xProbe2 SID 3 AVG	11248.54	32.66	669	64833.33	0

Table 8.4: SID 3 results

Scan	Duration in sec	Open	Closed	Filtered	No Information
NMap SID 4 1st run	29.60	13	34	0	0
NMap SID 4 2nd run	29.58	13	34	0	0
NMap SID 4 3rd run	29.48	13	34	0	0
NMap SID 4 AVG	29.55	13	34	0	0
Masscan SID 4 1st run	13	13	0	0	34
Masscan SID 4 2nd run	12	13	0	0	34
Masscan SID 4 3rd run	12	13	0	0	34
Masscan SID 4 AVG	12.33	13	0	0	34
xProbe2 SID 4 1st run	6225.02	12	34	1	0
xProbe2 SID 4 2nd run	9860.03	12	34	1	0
xProbe2 SID 4 3rd run	719.55	12	34	1	0
xProbe2 SID 4 AVG	5601.53	12	34	1	0

Table 8.5: SID 4 results

Scan	Duration in sec	Open	Closed	Filtered / Open	Unfiltered
NMap SID 5 1st run	6348.52	0	0	0	65535
NMap SID 5 2nd run	6355.67	0	0	0	65535
NMap SID 5 3rd run	8172.32	0	0	0	65535
NMap SID 5 AVG	6958.84	0	0	0	65535
NMap SID 6 1st run	6599.09	0	65487	48	0
NMap SID 6 2nd run	6600.53	0	65487	48	0
NMap SID 6 3rd run	6600.23	0	65487	48	0
NMap SID 6 AVG	6599.95	0	65487	48	0
NMap SID 7 1st run	6600.23	0	65487	48	0
NMap SID 7 2nd run	6599.29	0	65487	48	0
NMap SID 7 3rd run	6600.25	0	65487	48	0
NMap SID 7 AVG	6599.92	0	65487	48	0
NMap SID 8 1st run	6599.83	0	65487	48	0
NMap SID 8 2nd run	6598.08	0	65487	48	0
NMap SID 8 3rd run	6599.84	0	65487	48	0
NMap SID 8 AVG	6599.25	0	65487	48	0

Table 8.6: SID 5 6 7 8 results

	number of open ports	number of closed ports	number of filtered ports	hosts
Masscan SID 9 1st run	1152	1571688	-	-
Masscan SID 9 2nd run	1152	1571688	-	-
Masscan SID 9 3rd run	1152	1571688	-	-
Masscan SID 9 Average	1152	1571688	-	-
NMap SID 9 1st run	1152	1571688	-	24
NMap SID 9 2nd run	1152	1571688	-	24
NMap SID 9 3rd run	1152	1571688	-	24
NMap SID 9 Average	1152	1571688	-	24
xProbe2 SID 9 1st run	1152	1571663	25	24
xProbe2 SID 9 2nd run	1152	1571664	24	24
xProbe2 SID 9 3rd run	1152	15716556	32	24
xProbe2 SID 9 Average	1152	1571661	27	24

Table 8.7: SID 9 results

	number of open ports	number of closed ports	number of filtered ports	hosts
Masscan SID 10 1st run	312	816	-	-
Masscan SID 10 2nd run	312	816	-	-
Masscan SID 10 3rd run	312	816	-	-
Masscan SID 10 Average	312	816	-	-
NMap SID 10 1st run	299	782	-	23
NMap SID 10 2nd run	312	816	-	24
NMap SID 10 3rd run	312	816	-	24
NMap SID 10 Average	307.66	804.66	-	23.66
xProbe2 SID 10 1st run	288	816	24	24
xProbe2 SID 10 2nd run	288	816	24	24
xProbe2 SID 10 3rd run	288	816	24	24
xProbe2 SID 10 Average	288	816	24	24

Table 8.8: SID 10 results

	number of open ports	number of closed ports	number of filtered ports	hosts
NMap SID 11 1st run	299	782	-	23
NMap SID 11 2nd run	312	816	-	24
NMap SID 11 3rd run	286	748	-	22
NMap SID 11 Average	299	782	-	23

Table 8.9: SID 11 results

Scan	Duration in sec	Open	Closed	Filtered	No Information
NMap SID 12 1st run	123.96	24	0	0	0
NMap SID 12 2nd run	125.41	24	0	0	0
NMap SID 12 3rd run	123.96	24	0	0	0
NMap SID 12 AVG	124,443	24	0	0	0

Table 8.10: SID 12 results

Scan	Duration in sec	Open	Closed	Filtered	No information
NMap SID 13 1st run	120.43	24	0	0	0
NMap SID 13 2nd run	120.43	24	0	0	0
NMap SID 13 3rd run	91.38	22	0	0	2
NMap SID 13 AVG	110.476	23.33	0	0	0.66
Masscan SID 13 1st run	12	24	0	0	0
Masscan SID 13 2nd run	14	24	0	0	0
Masscan SID 13 3rd run	12	24	0	0	0
Masscan SID 13 AVG	11.33	24	0	0	0
ZMap SID 13 1st run	8	22	0	0	2
ZMap SID 13 2nd run	8	22	0	0	2
ZMap SID 13 3rd run	8	22	0	0	2
ZMap SID 13 AVG	8	22	0	0	2

Table 8.11: SID 13 results horizontal

Scan	Duration in sec	Open	Closed	Filtered	No information
NMap SID 14 1st run	75.45	0	22	0	2
NMap SID 14 2nd run	67.16	0	24	0	0
NMap SID 14 3rd run	67.16	0	23	0	1
NMap SID 14 AVG	69.923	0	23	0	1
ZMap SID 14 1st run	10.23	0	22	0	2
ZMap SID 14 2nd run	9.05	0	24	0	0
ZMap SID 14 3rd run	9.03	0	23	0	1
ZMap SID 14 AVG	9.44	0	23	0	1

Table 8.12: SID 14 results horizontal

OS detection run	Results
NMap 1st run	Netgear DG834G WAP / Western Digital WD TV media player (96%) Linux 2.6.32 (95%) Linux 2.6.32 - 3.9 (95%)
NMap 2nd run	Netgear DG834G WAP / Western Digital WD TV media player (96%) Linux 2.6.32 (95%) Linux 2.6.32 - 3.9 (95%)
NMap 3rd run	Netgear DG834G WAP / Western Digital WD TV media player (95%) Linux 2.6.32 (95%) Linux 2.6.32 - 3.9 (95%)
xProbe2 1st run	Linux Kernel 2.6.8 (96%) Linux Kernel 2.4.29 (92%) Linux Kernel 2.6.2 (92%)
xProbe2 2nd run	Linux Kernel 2.6.8 (93%) Linux Kernel 2.6.1 (90%) Linux Kernel 2.4.21 (90%)
xProbe2 3rd run	Linux Kernel 2.6.8 (93%) Linux Kernel 2.6.1 (90%) Linux Kernel 2.4.21 (90%)

Table 8.13: Result of the OS fingerprint tests with NMap and xProbe2

Scan	#alerts	#alerted ports	port range	alerted ports in %
NMap TCP Connect 1st run	1	100	1-100	100
NMap TCP Connect 2nd run	1	100	1-100	100
NMap TCP Connect 3rd run	1	100	1-100	100
NMap TCP Connect AVG	1	100		100
NMap SYN 1st run	1	100	1-100	100
NMap SYN 2nd run	1	100	1-100	100
NMap SYN 3rd run	1	100	1-100	100
NMap SYN AVG	1	100		100
NMap UDP 1st run	9	626	5-98	93
NMap UDP 2nd run	9	530	5-100	95
NMap UDP 3rd run	9	686	4-100	96
NMap UDP AVG	9	614		94.66
NMap ACK 1st, 2nd, 3rd run & AVG	0	0	0	0
NMap FIN 1st run	2	198	2-100	99
NMap FIN 2nd run	2	178	1-100	100
NMap FIN 3rd run	2	167	1-100	100
NMap FIN AVG	2	181		99.66
NMap XMAS 1st run	1	100	1-100	100
NMap XMAS 2nd run	1	99	1-99	99
NMap XMAS 3rd run	1	100	1-100	100
NMap XMAS AVG	1	99.66		99.66
NMap NULL 1st run	2	179	1-100	100
NMap NULL 2nd run	1	99	2-100	99
NMap NULL 3rd run	2	167	1-100	100
NMap NULL AVG	1.66	148.33		99.66
xProbe2 TCP 1st run	1	100	1-100	100
xProbe2 TCP 2nd run	1	100	1-100	100
xProbe2 TCP 3rd run	1	100	1-100	100
xProbe2 TCP AVG	1	100		100
xProbe2 UDP 1st run	0	0	0	0
xProbe2 UDP 2nd run	1	100	1-100	100
xProbe2 UDP 3rd run	1	100	1-100	100
xProbe2 UDP AVG	0.66	66.66		66.66
Masscan 1st run	1	100	1-100	100
Masscan 2nd run	2	136	1-100	100
Masscan 3rd run	2	132	1-100	100
Masscan AVG	1.66	122.66		100

Table 8.14: Results of PSAD towards scans from port number 1 up to 100

Scan	#alerts	#alerted port attacks	port range	alerted ports in %
NMap TCP Connect 1st run	0	0	0	0
NMap TCP Connect 2nd run	0	0	0	0
NMap TCP Connect 3rd run	0	0	0	0
NMap TCP Connect AVG	0	0	0	0
NMap SYN 1st run	1	87	9-96	87
NMap SYN 2nd run	1	74	13-87	74
NMap SYN 3rd run	1	59	21-80	59
NMap SYN AVG	1	73.33		73.33
NMap UDP 1st run	2	177	8-100	92
NMap UDP 2nd run	2	166	5-90	85
NMap UDP 3rd run	2	173	1-96	96
NMap UDP AVG	2	172		91
NMap ACK 1st, 2nd, 3rd run & AVG	0	0	0	0
NMap FIN 1st run*	100	100	1-100	100
NMap FIN 2nd run*	100	100	1-100	100
NMap FIN 3rd run*	100	100	1-100	100
NMap FIN AVG	100	100		100
NMap XMAS 1st run*	100	100	1-100	100
NMap XMAS 2nd run*	100	100	1-100	100
NMap XMAS 3rd run*	100	100	1-100	100
NMap XMAS AVG	100	100		100
NMap NULL 1st run	0	0	0	0
NMap NULL 2nd run	0	0	0	0
NMap NULL 3rd run	0	0	0	0
NMap NULL AVG	0	0		0
xProbe2 TCP 1st run	1	66	32-98	66
xProbe2 TCP 2nd run	1	79	14-93	79
xProbe2 TCP 3rd run	1	82	17-99	82
xProbe2 TCP AVG	1	75.66		75.66
xProbe2 UDP 1st run	1	84	2-86	84
xProbe2 UDP 2nd run	1	68	6-74	68
xProbe2 UDP 3rd run	1	66	1-78	66
xProbe2 UDP AVG	1	72.66		72.66
Masscan 1st run	1	82	11-93	82
Masscan 2nd run	1	77	20-97	77
Masscan 3rd run	1	71	25-96	71
Masscan AVG	1	76.66		76.66

Table 8.15: Results of Snort towards scans from port number 1 up to 100

Scan	#alerts	#alerted ports	port range	alerted ports in %
NMap TCP Connect 1st run	1	6	21, 80, 22, 23, 47, 95, ...,	6
NMap TCP Connect 2nd run	1	6	21, 25, 53, 23, 80, 22, ...,	6
NMap TCP Connect 3rd run	1	6	22, 53, 21, 23, 65, 61, ...,	6
NMap TCP Connect AVG	1	6		6
NMap SYN 1st run	1	6	23, 25, 53, 22, 21, 80, ...,	6
NMap SYN 2nd run	1	6	21, 53, 25, 22, 23, 80, ...,	6
NMap SYN 3rd run	1	6	23, 21, 22, 53, 80, 25, ...,	6
NMap SYN AVG	1	6		6
NMap UDP 1st run	0	0	0	0
NMap UDP 2nd run	0	0	0	0
NMap UDP 3rd run	0	0	0	0
NMap UDP AVG	0	0		0
NMap ACK 1st, 2nd, 3rd run & AVG	0	0	0	0
NMap FIN 1st run	1	6	53, 25, 80, 22, 23, 21, ...,	6
NMap FIN 2nd run	1	6	21, 25, 80, 53, 22, 23, ...,	6
NMap FIN 3rd run	1	6	21, 53, 22, 80, 25, 23, ...,	6
NMap FIN AVG	1	6		6
NMap XMAS 1st run	1	6	22, 53, 80, 25, 23, 21, ...,	6
NMap XMAS 2nd run	1	6	25, 21, 80, 53, 23, 22, ...,	6
NMap XMAS 3rd run	1	6	80, 53, 22, 23, 21, 25, ...,	6
NMap XMAS AVG	1	6		6
NMap NULL 1st run	1	6	22, 53, 23, 21, 25, 80, ...,	6
NMap NULL 2nd run	1	6	53, 22, 23, 25, 80, 21, ...,	6
NMap NULL 3rd run	1	6	53, 23, 21, 22, 25, 80, ...,	6
NMap NULL AVG	1	6		6
xProbe2 TCP 1st run	1	6	100, 68, 76, 87, 88, 57, ...,	6
xProbe2 TCP 2nd run	1	6	80, 31, 98, 23, 84, 46, ...,	6
xProbe2 TCP 3rd run	1	6	87, 88, 34, 39, 14, 2, ...,	6
xProbe2 TCP AVG	1	6		6
xProbe2 UDP 1st run	0	0	0	0
xProbe2 UDP 2nd run	0	0	0	0
xProbe2 UDP 3rd run	0	0	0	0
xProbe2 UDP AVG	0	0		0
Masscan 1st run	1	6	96, 67, 89, 83, 55, 80, ...,	6
Masscan 2nd run	1	6	30, 33, 14, 51, 46, 71, ...,	6
Masscan 3rd run	1	6	31, 8, 53, 23, 5, 80, ...,	6
Masscan AVG	1	6		6

Table 8.16: Results of Scanlogd towards scans from port number 1 up to 100

Scan	#alerts	#alerted ports	port range	alerted ports in %
NMap TCP Connect 1st run	1	8	80, 7, 15, 70, 79, 1, 9, 11	8
NMap TCP Connect 2nd run	1	8	80, 15, 9, 79, 11, 1, 7, 70	8
NMap TCP Connect 3rd run	1	8	9, 1, 80, 70, 7, 11, 15, 79	8
NMap TCP Connect AVG	1	8		8
NMap SYN 1st run	0	0	0	0
NMap SYN 2nd run	0	0	0	0
NMap SYN 3rd run	0	0	0	0
NMap SYN AVG	0	0		0
NMap UDP 1st run	1	6	67, 7, 9, 68, 1, 66	6
NMap UDP 2nd run	1	6	66, 7, 9, 67, 68, 1	6
NMap UDP 3rd run	1	8	1, 68, 1, 68, 7, 66, 67, 9	7
NMap UDP AVG	1	66.66		6.33
NMap ACK 1st, 2nd, 3rd run & AVG	0	0	0	0
NMap FIN 1st run	0	0	0	0
NMap FIN 2nd run	0	0	0	0
NMap FIN 3rd run	0	0	0	0
NMap FIN AVG	0	0		0
NMap XMAS 1st run	0	0	0	0
NMap XMAS 2nd run	0	0	0	0
NMap XMAS 3rd run	0	0	0	0
NMap XMAS AVG	0	0		0
NMap NULL 1st run	0	0	0	0
NMap NULL 2nd run	0	0	0	0
NMap NULL 3rd run	0	0	0	0
NMap NULL AVG	0	0		0
xProbe2 TCP 1st run	0	0	0	0
xProbe2 TCP 2nd run	0	0	0	0
xProbe2 TCP 3rd run	0	0	0	0
xProbe2 TCP AVG	0	0		0
xProbe2 UDP 1st run	1	6	67, 66, 9, 7, 68,1	6
xProbe2 UDP 2nd run	1	6	66, 1, 67, 7, 9, 68	6
xProbe2 UDP 3rd run	1	6	67, 7, 68, 66, 1, 9	6
xProbe2 UDP AVG	1	6		6
Masscan 1st run	0	0	0	0
Masscan 2nd run	0	0	0	0
Masscan 3rd run	0	0	0	0
Masscan AVG	0	0		0

Table 8.17: Results of Portsentry towards scans from port number 1 up to 100

Bibliography

[1] "Hackmageddon," accessed: 3.08.2015. [Online]. Available: http://www.hackmageddon.com/2015/08/03/16-31-july-2015-cyber-attacks-timeline/

[2] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions,Third Edition*, 3rd ed. McGraw-Hill Professional, 2001.

[3] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," 2003.

[4] "nmap.org/port scanning techniques," accessed: 20.05.2015. [Online]. Available: https://nmap.org/book/man-port-scanning-techniques.html

[5] "Rfc791," accessed: 05.06.2015. [Online]. Available: https://tools.ietf.org/html/rfc791

[6] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *J. Comput. Secur.*, vol. 10, no. 1-2, pp. 105–136, Jul. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=597917.597922

[7] N. Mäurer, "Efficient scans in a research network," Bachelorarbeit, Technische Universität München, München, Feb. 2015.

[8] N. E.-N. Kevin Daimi, "Evaluation of network port scanning tools," in *PROCEEDINGS OF THE 2011 INTERNATIONAL CONFERENCE ON SECURITY AND MANAGEMENT*. CSREA Press, 2011, pp. 465–472. [Online]. Available: http://www.gbv.de/dms/tib-ub-hannover/68892459x.pdf

[9] M. de Vivo, E. Carrasco, G. Isern, and G. O. de Vivo, "A review of port scanning techniques," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 41–48, Apr. 1999. [Online]. Available: http://doi.acm.org/10.1145/505733.505737

[10] R. Geng, "Project report of port scan detector," unknown. [Online]. Available: http://ruiligengstudy.com/research/security/Project%20Report%20of%20Port%20Scan%20Detector.swf

[11] S. Biles, "Detecting the unknown with snort and the statistical packet anomaly detection engine (spade)," unknown. [Online]. Available: http://webpages.cs.luc.edu/~pld/courses/447/sum08/class6/biles.spade.pdf

[12] Y. Chabchoub, C. Fricker, and P. Robert, "Improving the detection of on-line vertical port scan in ip traffic," in *Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on*, Oct 2012, pp. 1–6.

[13] A. George, P. Poornachandran, and M. R. Kaimal, "Adsvm: Pre-processor plug-in using support vector machine algorithm for snort," in *Proceedings of the First International Conference on Security of Internet of Things*, ser. SecurIT '12. New York, NY, USA: ACM, 2012, pp. 179–184. [Online]. Available: http://doi.acm.org/10.1145/2490428.2490453

[14] M. Dabbagh, A. Ghandour, K. Fawaz, W. Hajj, and H. Hajj, "Slow port scanning detection," in *Information Assurance and Security (IAS), 2011 7th International Conference on*, Dec 2011, pp. 228–233.

[15] "nmap.org," accessed: 17.05.2015. [Online]. Available: https://nmap.org

[16] R. Graham, "Masscan: the entire internet in 3 minutes," accessed: 20.03.2015. [Online]. Available: http://blog.erratasec.com/2013/09/masscan-entire-internet-in-3-minutes.html#.Vb_GXcDtlgp

[17] ——, "robertdavidgraham/masscan: Masscan: Mass ip port scanner," accessed: 20.03.2015. [Online]. Available: https://github.com/robertdavidgraham/masscan

[18] "Zmap," accessed: 21.03.2015. [Online]. Available: https://zmap.io/

[19] "Zmap documentation," accessed: 21.03.2015. [Online]. Available: https://zmap.io/documentation.html

[20] "X probe - active os fingerprinting tool," accessed: 01.04.2015. [Online]. Available: http://sourceforge.net/projects/xprobe/

[21] "Sentry tools," accessed: 16.03.2015. [Online]. Available: http://sourceforge.net/projects/sentrytools/

[22] "Portsentry for attack detection, part one," accessed: 16.03.2015. [Online]. Available: http://www.symantec.com/connect/articles/portsentry-attack-detection-part-one

[23] "psad: Intrusion detection and log analysis with iptables," accessed: 19.03.2015. [Online]. Available: http://cipherdyne.org/psad/

[24] "scanlogd - a port scan detection tool," accessed: 19.03.2015. [Online]. Available: http://www.openwall.com/scanlogd/

[25] "Snort," accessed: 24.03.2015. [Online]. Available: https://snort.org/

[26] “2.2.4 sfportscan,” accessed: 24.03.2015. [Online]. Available: http://manual.snort.org/node17.html#SECTION00324000000000000000

[27] “Chapter 15. nmap reference guide,” accessed: 17.05.2015. [Online]. Available: https://nmap.org/book/man.html

[28] “Scanlogd,” accessed: 03.04.2015. [Online]. Available: http://www.openwall.com/scanlogd/scanlogd.8.shtml

[29] “Readme.sfportscan,” accessed: 24.03.2015. [Online]. Available: https://github.com/vrtadmin/snort-faq/blob/master/docs/README.sfportscan