**УНИВЕРЗИТЕТ „Св. Кирил и Методиј“**

**ПРИРОДНО МАТЕМАТИЧКИ ФАКУЛТЕТ - СКОПЈЕ**



**DEVELOPING OPTIMAL MODELS OF RULE SETS FOR INTRUSION DETECTION SYSTEMS**

**- дипломска работа -**

Кандидат,Ментор,

Александар Христовски Проф. д-р Ванчо Чабуковски

Скопје, 2018

***Abstract:*** *At present times, computers and computer networks are ever present in all aspects of life, thus, the field of computer and network security is of paramount importance. Oftentimes, the problem of computer and network security hangs on optimal use of available resources. Resources, in this sense of the word, entail hardware, software as well as personnel trained in network management and monitoring. Intrusion detection systems are part of overall network security systems and are primarily responsible for continuous network monitoring for potential threats. If a threat is found, the system should alert the people responsible to it. Dealing with such threats in a timely fashion could mean the difference between uninterrupted workflow and complete network failure bearing capital and trust losses. Like most systems, intrusion detection systems are only as good as their configuration. Intrusion detection is usually performed by testing whether network packets conform to a defined set of rules. If they do not, the IDS regards them as potentially hostile anomalies. It is a common practice for network administrators to use all, or at least a large subset of the predefined rules that come with the IDS. Such a practice can lead to excessive packet rejection, analysis skipping a portion of the packets, or a large number of false positives. Therefore, it is of great importance that the administrator understands the network requirements regarding an IDS, and configure the rule set appropriately. This paper lays out certain principles and guidelines that would lead to optimal configuration of intrusion detection systems.*

***Keywords:*** *Computer security, network security, IDS, intrusion detection systems*

Содржина

[1. Вовед 5](#__RefHeading___Toc189_3320784199)

[2. Мрежни protocolи 11](#__RefHeading___Toc191_3320784199)

[2.1 IP (Internet Protocol) 11](#__RefHeading___Toc193_3320784199)

[2.2 TCP (Transmission Control Protocol) 14](#__RefHeading___Toc197_3320784199)

[2.3 UDP (User Datagram Protocol) 18](#__RefHeading___Toc203_3320784199)

[3. Техники и алатки за напад 20](#__RefHeading___Toc207_3320784199)

[3.1 nMap 24](#__RefHeading___Toc209_3320784199)

[3.2 Nessus 24](#__RefHeading___Toc211_3320784199)

[3.3 Metasploit 25](#__RefHeading___Toc213_3320784199)

[4. Техники и алатки за откривање и заштита од напади 26](#__RefHeading___Toc215_3320784199)

[4.1 Firewall 26](#__RefHeading___Toc217_3320784199)

[4.2 IDS/IPS 27](#__RefHeading___Toc219_3320784199)

[4.3 Snort 29](#__RefHeading___Toc221_3320784199)

[4.4 Suricata 30](#__RefHeading___Toc223_3320784199)

[5. Тестирање и анализа за оптимални IDS правила за различни типови на scan 31](#__RefHeading___Toc225_3320784199)

[5.1 Откривање на хостови 31](#__RefHeading___Toc227_3320784199)

[5.2 scan на порти 36](#__RefHeading___Toc229_3320784199)

[5.3 No Ping Scan и генерирање на оптимални IDS/IPS правила 43](#__RefHeading___Toc231_3320784199)

[5.4 TCP Connect Scan и генерирање на оптимални IDS/IPS правила 44](#__RefHeading___Toc233_3320784199)

[5.5 Xmas Scan и генерирање на оптимални IDS/IPS правила 46](#__RefHeading___Toc235_3320784199)

[5.6 FIN Scan и генерирање на оптимални IDS/IPS правила 48](#__RefHeading___Toc237_3320784199)

[5.7 NULL Scan и генерирање на оптимални IDS/IPS правила 49](#__RefHeading___Toc239_3320784199)

[5.8 UDP Scan и генерирање на оптимални IDS/IPS правила 51](#__RefHeading___Toc241_3320784199)

[5.9 SYN Scan и генерирање на оптимални IDS/IPS правила 52](#__RefHeading___Toc243_3320784199)

[6. Креирање на оптимален модел на IDS правила за различни мрежни околини 55](#__RefHeading___Toc245_3320784199)

[6.1 Мрежи на мали и средни организации и соодветни правила 55](#__RefHeading___Toc247_3320784199)

[6.2 Мрежи на IT организации и соодветни правила 58](#__RefHeading___Toc463_3320784199)

[6.3 Мрежи на средни и големи организации и соодветни правила 60](#__RefHeading___Toc251_3320784199)

[7. Заклучок 64](#__RefHeading___Toc781_3320784199)

[7. Користена литература 66](#__RefHeading___Toc675_3320784199)

**Листа на слики**

Figure 1: Minimal security network

Figure 2: High security network

Figure 3: IP protocol header

Figure 4: TCP protocol header

Figure 5: UDP protocol header

Figure 6: No ping scan

Figure 7: Snort log of No Ping scan

Figure 8: Suricata log of No Ping scan

Figure 9: TCP Connect scan

Figure 10: Snort log of TCP Connect scan

Figure 11: Suricata log of TCP Connect scan

Figure 12: Xmas scan

Figure 13: Snort log of Xmas scan

Figure 14: Suricata log of Xmas scan

Figure 15: FIN scan

Figure 16: Snort log of FIN scan

Figure 17: Suricata log of FIN scan

Figure 18: NULL scan

Figure 19: Snort log of NULL scan

Figure 20: Suricata log of NULL scan

Figure 21: UDP scan

Figure 22: Snort log of UDP scan

Figure 23: Suricata log of UDP scan

Figure 24: SYN scan

Figure 25: Snort log of SYN scan

Figure 26: Suricata log of SYN scan

Figure 27. Sample network of a small organization

Figure 28. Sample network of an IT organisation

Figure 29. Sample network of a large organisation

# 1. Introduction

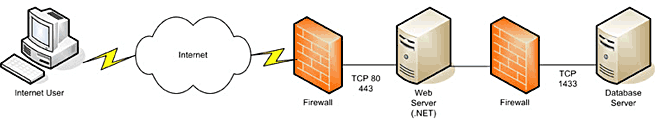
Information security is a field concerned with confidentiality, integrity и availability of information [2] (the abbreviation CIA – Confidentiality, Integrity, Availability is the most popular term encapsulating the key concepts of information security). Although information security takes into regard the security of data written on any medium, today this term is almost synonumous with the term computer security. In the context of computer security, confidentiality means ensuring that unauthorized parties do not get access to certain information. By extension, this definition covers everyday situations such as leaving a laptop at a public location, or sending an email containing sensitive data to a wrong address in addition to highly technical network breaches. Integrity, the second key concept of information security entails maintaining the consistency, accuracy and trustworthiness of data over its life cycle usually by preventing different parties from altering the data in an unnoticeable manner. Public key infrastructure, or PKI offers hashing, Message Authentication Codes (MAC) and other crucial tools to achieving data integrity. For the data to be useful, it has to be available when needed. The concept of availability involves keeping and maintaining the computer systems that work with the data in an impeccable state, as well as their security components, and are capable of preventing or mitigating service interruptions caused by anything from Denial of Service attacks to power outages and natural disasters.

Lately, the concept of non-repudiation is oftentimes added to the array of key concepts of information security. Non-repudiation is a concept that originates in law, and guarantees that a party that sends a certain message cannot later deny sending that message, and also if this message is transferred successfully, non-repudiation prevents the receiver from denying they received the message. Similar to the concept of data integrity, non-repudiation uses tools provided by Public Key Infrastructure.

Parkerian Hexad is a security model conceived by Donn B.Parker in 1998 that adds three additional attributes to the CIA triad, namely possession (control), authenticity and utility. Possession is a concept applied when contemplating potential damage inflicted by a loss in confidentiality. Authenticity is a concept similar to integrity, but is specifically concerned with proving authorship of a certain message. Usability is an extension on the concept of availability. A commonly used example explaining usability is having access to the needed data in it’s encrypted form, but not having the decryption key, therefore, the information is available, but not usable.

In a corporative environment, jobs concerned with information security are as follows: Chief Security Officer, Chief Information Security Officer, security engineer, security analyst, secure systems administrator, as well as computer security consultant.

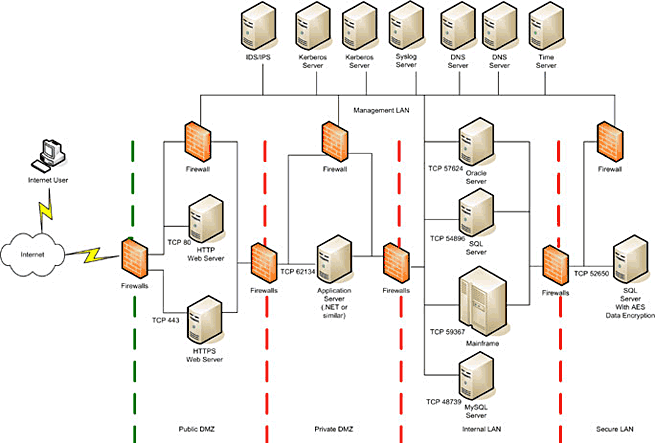
Securing a network is oftentimes a question of finances that can be attributed to this end. A firewall is one necessary fundamental component of a secure network. In the classical, simple case where a network of a single web server and a database server is to be secured, the minimal acceptable solution involves setting up a firewall at each of the critical points of protection, one preceding the web server and another preceding the internal network as shows in Figure 1.

***Figure 1. Minimal security network***

In this relatively cheap setup, an attacked would only have to compromise a single server before accessing the web applications running on the network. However, network firewalls like the ones in this scenario cannot offer protection against attacks on an application levels such as buffer overflows or injections. In addition, the efficiency of the firewall hangs upon the efficiency of it’s configuration, and the financial cost of such a system rises by the amount of money spent on the administrator providing that configuration and maintaining the firewalls.

As a general rule, the security of a certain network is incremented by incrementing the number of distinct steps that a potential attacker would have to perform in order to compromise the network. Techniques used in securing a network today include implementing a demilitarized zone (DMZ, also known as a perimeter defense) such that it would expose the network services to external, untrusted networks. A DMZ can be further divided into a public and a private zone, where the public zone would contain the user interface logic, while the private zone would contain the business logic of applications. In addition to DMZs, a secure network would also have:

* IDS, intrusion detection systems, commonly NIDS (Network Intrusion Detection System), systems that would monitor traffic in critical areas of both the DMZ and secure LAN.
* Time Server, usually used for synchronizing logging services.
* Logging services, or SIEM (Security Information and Event Management), used for event information collection from network critical systems, such as firewalls, routers, IDSs etc.
* Firewalls, which’s properly written rules are of paramount importance.
* Internal Domain Name Servers, configured such that, all packets designated out of the network are delegated to the Internet Service Provider’s DNS. This would enable the network administrator to closely monitor the DNSs which could prevent attacks like DNS poisoning.
* Secure storage servers, used to store data, should abide by two key rules, namely encrypting everything that is stored and limiting the number of parties that have access to data.
* Kerberos servers, providing authentication to all application servers.



***Figure 2. High security network***

Usage of IP subnets where appropriate, Virtual LANs and virtual machines in combination with Host based IDS are all recommended additional security steps.

Intrusion Detection Systems, or specifically their rule sets are the main focus of this paper. Usage of all or vast majority of the predefined rules these systems ship with is common, and writing network-specific rules is often neglected.

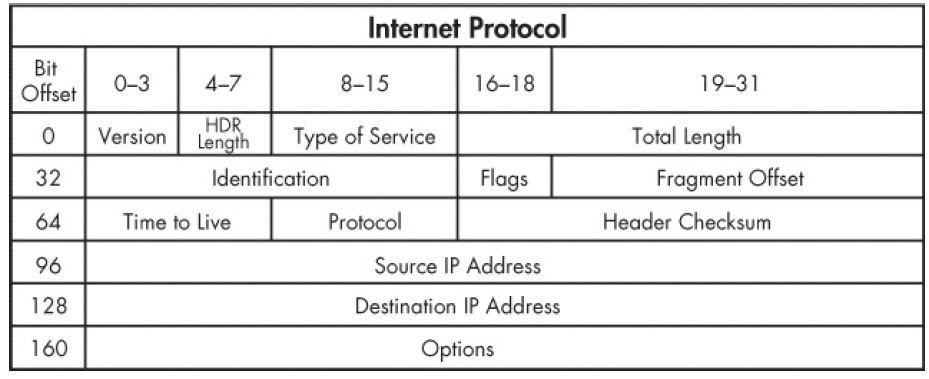
By writing such rules while take the specific requirements of a certain network into account, one could significantly increase the efficiency of an IDS. Further down in this paper several generic cases are used to develop guidelines for writing IDS rules.

# 2. Network protocols

## 2.1 IP (Internet Protocol)

Internet Protocol is the primary protocol enabling communication via the internet. IP’s primary concern is transporting datagrams through network borders [5]. When they are in the same network, devices can communicate by using hardware (MAC) addresses, but when one of them in out of the collision space (behind a router or a switch) communication requires the usage of IP addresses. If a packet is to be sent out of the local network, it is encapsulated in an IP datagram and is set to the default gateway, which is usually a router. This router using tables and prefixes makes a decision about the port it should resend the packet through, and if no appropriate port is found, the packet is resent to it’s default gateway, and this process reiterates on a series of devices until the packet reaches it’s destination network, is unpacked and sent to it’s intended recipient.

### IP Header

Figure 3 shows  the IP protocol header:

***Figure 3. IP protocol header***

* Version – Designates the Internet Protocol version (4 or 6)
* IHL, Internet Header Length – designates the header length in 32 bit words. Minimal header size is 20 bits, and maximal header side is 160 bits.
* Differentiated Services Code Point/Type of Service – A field mainly utilized by newer services which require synchronized data, such as Voice Over IP.
* Explicit Congestion Notification is a field meant to indicate the degree of current network usage, which is optional and used only when all parties to the communication support it.
* Total Length – designates the size of the complete size of the packet, header and data included. Minimal value of the total length is 20 bits, and maximal value 65.635.
* Identification – a field used to identify fragment group in an IP datagram.
* Flags – holds values for the 3 IP flags, specifically:

1. bit 0: Reserved, always set to 0
2. bit 1: Don’t Fragment (DF)
3. bit 2: More Fragments (MF)

If the DF field is set to 1 and fragmentation is necessary the packet is dropped.

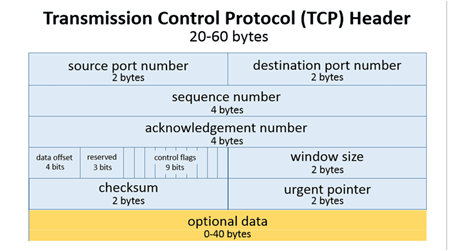
* Fragment Offset – holds the offset of a certain fragment from the original, not fragmented IP datagram
* Time To Live – is a field which’s value is decremented with each router hop, and is meant to prevent a situation where the packet is circulating between routers in an infinite loop, adding unnecessary strain to the network. If the value of the TTL field reaches zero, the packet is dropped. In practice it’s commonly used as a hop count, i.e. for keeping track of the number of routers the packet goes through before reaching it’s destination.
* Protocol – a field designating the IP datagram’s underlying protocol code.
* Header Checksum – holds a 16 bit IP header checksum value which is recalculated and rechecked by each router traversed by the datagram. In case of a checksum mismatch the packet is discarded. Since each router decrements the value of the TTL field, the checksum changes on each hop.
* Source Address – a field containing the source address of the packet. It’s value is changeable due to NAT.
* Destination Address – a field containing the destination address of the packet. Like the source address, this field can be changed because of NAT.
* Options – a field that is almost never put to use.

## 2.2 TCP (Transmission Control Protocol)

Transmission Control Protocol is a communications standard found in the Transport layer of the OSI model [5]. TCP is a connection oriented protocol that offers reliable information exchange between the communicating parties.

### TCP Header

Figure 4 depicts the TCP header which is between 20 and 60 bytes in size:



***Figure 4. TCP protocol header***

* Source port — 16 bits designating the source port number
* Destination port — 16 bits designating the destination port number
* Sequence number — 32 bits designating the sequence identification number
* Acknowledgement number — 32 bits designating the ACK identification number
* Data Offset — 4 bits, specifying the header length, expressed in 32-bit words
* Reserved — 3 bits, meant for a possible future use, set to zero
* Контролни знаменца — 9 bits of which six are in use, specifically:  
  1. URG — indication whether the Urgent pointer should be considered
  2. ACK — indicating whether the Acknowledgement field should be considered. Set to 1 in all packets sent, except the initial one.
  3. PSH — when set to 1, the PSH flag tells the receiving computer to pass on the current buffer to the receiving application.
  4. RST — when a packet with the RST flag set to 1 is received the connection should be reset.
  5. SYN — only the initial packet should have this flag set to 1. It is used for synchronizing sequence numbers
  6. FIN — designating the final packet sent from the communicating party.
* Window size — 16 bits specifying the window size i.e. the number of bytes that should be sent before receiving an ACK packet.
* Urgent pointer — 16 bits designating an urgent portion of the packet. Only used in case the URG flag is set to 1.
* Options — field containing between 0 and 320 bits used in setting up options such as maximal segment size, window size, selective requiring ACK responses, timestamps etc. Commonly this part of the header is set to 0 and default options are used.

### TCP connection

Steps involved in establishing a TCP connection are as follows:

1. Initiating party sends a SYN packet with an arbitrarily chosen sequence number
2. The other communicating party sends the initiating party a SYN ACN packet. The ACK field in this packet contains the expected following SYN number, which is the current sequence number incremented by one, while the sequence field in this packet is set to the current sequence number.
3. The initiating party send a packet with no data payload and an ACK number copied from the preceding packet. Since there is no data in this packet the receiving party does not increment it's sequence number. The connection is now established.
4. Data transfer now occurs via the established connection with each party incrementing their respective sequence and ACK numbers expressed in bytes with each packet.
5. Following the data transfer, one of the parties might want to close the connection and send a FIN packet with the appropriate sequence number (equivalent to the ACK number of the preceding packet).
6. After receiving a FIN packet a communicating party replies with an ACK packet followed by their own FIN packet with the appropriate sequence number.

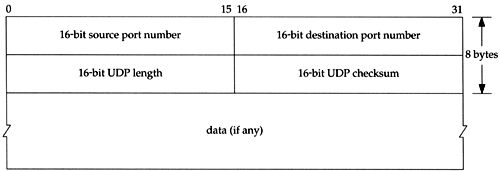
After receiving the FIN packet sent in step 6, the party that initiated the closure send and ACK packet back to the other party with a sequence number incremented by one and the connection is successfully closed.

## 2.3 UDP (User Datagram Protocol)

User Datagram Protocol is considered one of the fundamental internet protocols. Its formal description is written by David P. Reed in 1980. UDP is a connection-less protocol offering a minimalistic approach with very little overhead that does not include a handshaking mechanism, error detection or error correction and therefore is used when non-reliable, error prone, but fast transfer is appropriate [5]. Use cases include, but are not limited to: services streaming audio or video over the internet, DNS, Network Time Protocol, IP tunneling, Remote Procedure Call, Network File System, DHCP, Trivial File Transfer Protocol, UPTV, VoIP, online games, Routing Information Protocol.

### UDP Header

1. The UDP header contains four distinct fields at two bytes each (figure 5):



***Figure 5. UDP protocol header***

* Source port – if a need arises it is assumed that the receiving party can reach the source application at this port. If this is not possible the source port field should be set to zero.
* Destination port – an ephemeral port if the destination is a client of the application, or a well-known port in case the destination is the application server.
* Length – designating the UDP datagram size, header and data payload included expressed in number of bytes. This field can have a minimal value of 8, and a maximal value of 65507 if the protocol is run over IPv4, and larger if it is run over IPv6.
* Checksum – used for error detection in the header or data payload. Set to zero if not applicable.

# 3. Attack techniques and tools

Computer attack can be categorized by various criteria [1]. Taking legality and motivation of the attack yields:

* white hat, otherwise known as penetration testing attacks. These are completely legal and done in agreement with the attacked party in order to test the security of a network
* black hat, attacks which are illegal and can be brought about by a plethora of motives including profit, activism, curiosity etc.
* gray hat, attacks meant to point out a vulnerability at a certain network to it’s owners. While well-intentioned, grey hat attacks are not performed in agreement with the attacked party and are therefore illegal.

Taking the attackers identity into consideration we can differentiate between:

* computer act of war, where the attacked and defender are both nations or states
* computer crime, where the attacking party is an individual or an organization
* computer terrorism, where, again the attacking party is an individual or an organization

Regardless of the classification, all attacks follow the same pattern of finding and exploiting a vulnerability. To this end the attacker can use pre-made or write their own tools. Writing their own tools would require a fundamental understanding of the attack vector and it’s underlying principles, and furthermore the existence of well-written tools for most of the phases in an attack would render writing new ones obsolete.

An attack against a certain network usually spans over several months and can be divided into seven discrete phases [2]:

1. Reconnaissance

In this phase the attacker goes over the publicly available information regarding the target organization. Information gathered with reconnaissance include web servers addresses, mail server addresses and similar information obtainable by social engineering and other non-intrusive research.

1. Scanning

Scanning is the first intrusive phase in an attack. Here, the attacker tries to find an exploitable “weak spot”. This is the first undertaken action that can be detected by the defender. We can furthermore divide scanning by functionality into host discovery, which would inform the attacker of the active hosts in a network, and port scanning, which might reveal host’s open ports, services running on the target network and their vulnerabilities. nMap is a well known tool that has become the de-facto standard scanning software, while Nessus and SAINT are two of the most commonly used vulnerability scanners.

1. Gaining access and escalating privileges

Assuming the attacker has found a vulnerability in the previous step, at this point they exploit that vulnerability to gain “presence” on the target network. Since the goal of an attack almost always requires elevated privileges on the target network, an attack is not likely to stop at mere presence. Escalating privileges is a phase that usually involves exploiting vulnerabilities, password cracking and social engineering. Техниките користени во оваа фаза се експлоатирање на најдени ранливости, пробивање на лозинки и социјален инжињеринг. Metasploit is a well-known software packet which automizes large portions of network exploitation and it’s usage can be invaluable to the attacker at this point, although in an ideal scenario the attacker would be able to find and exploit an undocumented vulnerability, which would significantly diminish the odds of the unauthorized access being noticed by the target network’s security mechanisms. However, this is seldom possible.

1. Exfiltration

Exfiltration is a process of unnoticeable data transfer from the target network to a system owned by the attacker. This can oftentimes be achieved by simply copying the data to another server, although sometimes an ad hoc approach is warranted.

1. Maintaining access

After gaining presence and sufficient privileges on the target network the attacker, oftentimes would want to ensure continuous undetected access. Tools like root kits, trojan horses can provide this and many such software packets are also included into the Metasploit framework, although not documented tools authored by the attacker might prove more difficult to be found by the networks security system.

1. (Destructive) Attack

In this is an optional step the stealthiness of the attack stops being a factor, and the goal is inflicting a maximal amount of damage to the target network and disrupting the services offered by it.

1. Erasing traces

At this point the attacker would want to erase all traces of the attack and keep his presence a secret. Common steps undertaken in this phase are cleaning system logs, IDS logs as well as ad hoc or network specific actions.

## 3.1 nMap

nMap (“Network Mapper”) is a free, open source host discovery and port scanner [6]. It uses raw IP packets, as well as advanced scanning techniques that make nMap very effective for mapping active hosts, even those that reside behind IP filters, firewalls or routers. Nmap also offers various scanning tools (based on various protocols including TCP and UDP), operating system detection, software version detection, ping sweep and more.

## 3.2 Nessus

Nessus is the most commonly used vulnerability scanner in the world. Scanning with this software can reveal:

* Vulnerabilities that can be used by an attacker to gain access and control over a certain system.
* Non-secure software configuration.
* Usage of default passwords, common password, or lack of passwords.
* Possibility of a Denial Of Service attack by crafting special packets on a specific TCP/IP implementation.

Beyond classical penetration testing, Nessus oftentimes find use in the preparation of a certain network for offering financial services. The process employed by Nessus entails port scanning and automated running of exploits against open ports. One of the reasons for Nessus’s fame is it’s constantly maintained exploit database.

## 3.3 Metasploit

The Metasploit Framework is one of the most well-known projects in the field of computer security due to its vast exploit database, vulnerability scanners, remote administration tools, trojan horses, root kits and more. Using Metaspolit an attack could:

* Choose an exploit from its collection of over 1677 exploits for various operating systems.
* Optionally check if the target system is vulnerable to the chosen exploit.
* Choose a payload to be executed on the target computer, should the exploitation be successful.
* Choose an encoding technique such that the payload would run undetected by the target’s IDS/IPS.
* Perform all these steps using Metasploit commands.

In addition, the Metasploit Framework contains a packet called “Command shell” used to execute commands on the target computer, a packet called “Meterpreter” which can be used to transfer data or open a VNC with the target as well as options for creating dynamic code for trojan horses, viruses and root kits which would potentially increase their chances of remaining undetected.

# 4. Techniques used for attack detection and protection against attacks

## 4.1 Firewall

Firewall is a security system meant to control incoming and outgoing traffic. Traditionally, firewalls inspect packet headers and run them against a rule set to determine legal from malicious traffic, discarding packets not conforming to the rule set [1]. Lately there has been an emergence of Next Generation Firewalls (NGFWs), firewalls that can also serve as Intrusion Prevention Systems (IPSs). By looking at the way they function, firewalls can be classified into:

* packet filters, working on the second and third layer of the OSI model. These firewalls differentiate malicious packets to be discarded from normal traffic based on their header
* stateful firewalls that mind the state each connection is in, and therefore are capable to make more “intelligent” differentiation between normal and malicious traffic. Stateful firewalls operate on the sixth layer of the OSI model.
* proxy firewalls are dedicated devices placed at the network gateway which monitor all traffic and resend non-malicious traffic. These firewalls work on the application layer of the OSI model.

Modern firewalls usually fall into more then of the aforementioned classes. Firewalls are a fundamental and indispensable security tools, and as such hold a vast portion of the market of security devices. Some of the most commonly used firewalls are Fortinet FortiGate, Cisco ASA, Sophos UTM, pfSense.

## 4.2 IDS/IPS

IDS (Intrusion Detection System), is a hardware of software device meant to monitor traffic on a certain network (NIDS, Network IDS) or a single computer (HIDS, Host IDS). Host based IDSs mostly monitor the input and output streams of the storage. NIDS perform detection based on information gathered from the header and data payloads of IP packets and are capable of differentiating normal from malicious activity. Upon discovery of malicious packets the IDS notifies the administrator usually by logging the packet with an alert message info a SIEM, Security Information and Event Management. Differentiating “good” from “bad” traffic is done via either using a provided rule set, or by a process called anomaly detection.

In the case of differentiation by rule sets the network administrator is supposed to disseminate what they would consider “normal” traffic into formal rules and configure those rules into the IDS. Therefore the IDS would be able to alarm the administrator for potentially harmful traffic by catching packets that do not conform to the rule set. Such a system is only as effective as it’s configuration, and cannot detect novel approaches to an attack. Snort and Suricata are classical examples of rule based IDSs, with Suricata also doubling as an Intrusion Prevention System.

IDSs which utilize traffic differentiation by anomaly detection learn what the networks “normal” traffic is by machine learning methods and heuristics on a specific “training” set. If the “training” proves successful, such a system could notify the administrator of any deviation from the standard traffic that occurs. Anomaly detection holds the advantage of detecting novel attacks, however it’s drawbacks include a large percentage of false positives and the difficulty of providing an appropriate training set. The best knows IDS that works with anomaly detection is Bro.

Intrusion Prevention Systems, also known as active IDSs, are in fact Intrusion Detection Systems with the added ability of discarding packets that do not conform to the configured rules. The primary difference between IPSs and firewalls is in the subject of their analysis. Firewalls are limited to analyzing packets headers, while IPSs look at headers and data payloads. With the emergence of Next Generation Firewalls this difference is dissipating [3]. Well known IPSs include Suricata, IBM Security Intrusion Prevention System, Cisco FirePower and others.

## 4.3 Snort

Snort can take the role of a packet sniffer (tcpdump equivalent), packet logger, useful in debugging network data flow and discovery of bottlenecks as well as a full IDS. Originally created by Martin Roesh in 1998, Snort is currently developed by Sourcefire, a company whose founder is Roesh and is owned by Cisco.

Snort includes a simple, yet effective language for writing rules. A generic Snort rule follows this pattern:

action protocol external-address external-port ←direction→ internal-address internal-port options

Here, action tells Snort what to do in case a certain type of packet is detected; external-address and external-port are an address and a port found outside the monitored network; internal-address and internal-port designate a location in the monitored network; direction tells Snort whether the administrator is interested in an incoming or outgoing packet; options usually holds the message Snort is supposed to log, the rule ID, revision count, as well as specific data payload checks.

Example:

log tcp 192.168.0.52/24 any → 192.168.0.24 22 (msg: “internal SSH access”;)

tells Snort to log the message “internal SSH access” each time it detects TCP packets originating from the local network sent to port 22 on 192.168.0.24, while the rule:

alert tcp !192.168.0.52/24 any → 192.168.0.24 22 (msg: “external SSH access”;)

tells Snort to write an alert saying “external SSH access” each time it detects a TCP packet originating from outside the network sent to port 22 on 192.168.0.24.

## 4.4 Suricata

Suricata is an open source IDS/IPS. Emerging in July 2010, Suricata is continuously developed by the Open Information Security Foundation (OISF).

The rule language used by Suricata is almost identical to the one used by Snort, however Suricata’s language supports additional actions used in dropping packets.

# 5. Testing and analysis of optimal IDS rules for different scan types

## 5.1 Host Discovery

Host discovery is a process meant to determine active hosts from a larger set of addresses [6]. Methods used in host discovery include:

* List Scan (-sL) is a primitive method of host discovery which works by running reverse DNS resolution on all listed addresses. Results can be useful to determine which hosts belong to the target network, and an attacker could extrapolate additional information based on host names. E.g. if an attacker is to target PMF, and the List Scan results include a host name fw.pmf, the attack could extrapolate that that host is the firewall. Since this scan does not require packets sent directly to the scanned hosts, it is regarded as a stealth scan, however this scan type cannot tell the attacker anything more then a list of host names.
* No Port Scan (-sn), (a.k.a. ping scan, -sP in earlier nMap versions) that serves the same purpose as the List Scan, however, No Port Scan sends packets to the target hosts instead of just performing reverse DNS resolution on the target address list. Also, unlike the List Scan, this method only lists reachable hosts and can be used as a quick but relatively silent reconnaissance tool that can be augmented with traceroute and additional nMap scripts. By default the No Port Scan sends an ICMP Echo Request (ping), TCP SYN packet to port 443, TCP ACK packet to port 80 and an ICMP timestamp request. While scanning a local ethernet network ARP requests are used unless the –send-ip option is used.
* No Ping (-Pn) (-P0 or -PN in earlier nMap versions) circumvents host discovery entirely and runs a port scan, version detection and all scans required directly on all available, listed hosts. On local ethernet networks ARP packets are used by default, and this behavior can be changed by –disable-arp-ping or –send-ip flags.
* TCP SYN Ping (-PS <port-list>) sends an empty TCP SYN packet to port 80 by default. If different ports are to be used, they should be listed immediately after the -PS option. E.g. executing -PS22-25,80,113,1050,35000 would send the empty TCP SYN packet to ports 22, 23, 24, 25, 80, 113, 1050, 35000 on the target addresses. The SYN flag being set to 1 tells the target computer that the computer running the scan wants to establish a TCP connection. If the packet is sent to a closed port, the target computer will reply with a RST packet, otherwise the reply would be a SYN/ACK packet, and the kernel of the computer running the scan would send a RST, since it would consider the SYN/ACK to be unsolicited. In network discovery both RST and SYN/ACK replies would tell nMap that the host is active.
* TCP ACK Ping (-PA <port-list>) is a ping scan, similar to the TCP SYN scan, with the difference that an empty ACK packet is used by nMap instead of the SYN packet in the previous case. Just like the TCP SYN Ping port 80 is used by default and this can be overridden by listing the desired ports immediately after the -PA flag. Since the ACK flag is set to 1, an active host should always consider the packet as unsolicited and reply with a RST. Combining both -PS and -PA should maximize the odds of a packet being allowed through a firewall. A configuration often used by firewalls (and available by default on Linux Netfilter/iptables) blocks all SYN packets meant for any address except publicly available services commonly found on web or mail servers. In such a case the TCP ACK Ping scan would be of use, while the TCP SYN Ping scan packets would be dropped. Stateful firewalls would block packets that do not conform to the current connection state, which would lead to the opposite situation.
* UDP Ping (-PU <port-list>) sends an empty UDP packet to port 40125 to the target computer, unless a different port is specified. Having protocol-specific data might increase the chances of getting a reply on certain ports. Packet content can be set by using the –data, --data-string or –data-length options. If the packet is sent to a closed port an ICMP Port Unreachable reply would be received in response. This would indicate an active host, while other ICMP packets such as Host Unreachable/Network Unreachable, TTL exceededАко се наиде на затворена порта на хостот се добива ICMP Port Unreachable пакет назад. Ова сигнализира дека хостот е активен, додека друг тип на ICMP пакет како host/network unreachable или TTL exceeded or lack of a response would indicate that the address is not in use. If the port is open, most running services would ignore the packet and not send a reply. That is why 40125, a port that is almost never in use is the used by default. The main advantage of the UDP scan is that UDP packets could be allowed through a TCP packet filter.
* SCTP INIT Ping (-PY <port-list>) is a scan which utilizes SCTP INIT packets, i.e. the first step of the SCTP four-way handshake mechanism. If no different port is specified, port 80 is used. A closed port would result in an ABORT reply, and an open one in INIT-ACK. Either one would indicate an active host.
* ICMP Ping Types (-PE, -PP, -PM) make use of the standard ping tool in order to perform host discovery. The -PE flag is used to send an ICMP type 8 (echo request), in expectation of a type 0 (echo reply). Many modern hosts and firewalls block such requests (in spite of RFC 1122), therefore ICMP scanning is considered to be unreliable for scanning over the internet, however ICMP Ping Scan is still used to monitor local networks. ICMP timestamp and ICMP subnet requests can be utilized by using the -PP and -PM flags accordingly.
* IP Protocol Ping (-P0 <protocol-list>) sends protocol-specific packets for each of the protocols specified in the protocol list. By default, ICMP(protocol 1), IGMP(protocol 2) and IP-in-IP(protocol 4) packets are sent. Oftentimes TCP(protocol 6), UDP(protocol 17) and SCTP(protocol 132) are added to this list. In reply, the IP Protocol Ping expects either a packet of a specific protocol, or an ICMP Protocol Unreachable error. Both would mean the host is active.
* ARP Ping (-PR) is a ping scan often used in scanning local ethernet networks. Most LANs, especially ones using ranges of private addresses have a large portion of unused addresses at a given time. By sending an ICMP Echo Request for example nMap forces the operating system to find the destination hardware (MAC) address. This could prove problematic since the corresponding operating system libraries are not meant to support a large number of ARP requests at once. ARP Ping utilizes the optimized nMap libraries instead of the OS’s. If a reply is received, nMap knows the host is active. Over IPv6 -PR utilizes Neighbor Discovery, the IPv6 ARP equivalent.
* --disable-arp-ping tells nMap to avoid using ARP or ND packets in network discovery.
* --traceroute is a flag providing a combination of an address and port most likely to reach the target host. This is achieved by starting with using packets with a low TTL, then gradually incremented until a reply different from ICMP Time Exceeded reply is received.
* -n му кажува на nMap никогаш да не прави reverse DNS резолуција на активните IP адреси, и може значително да го скрати времето потребно за scan.
* -R prevents nMap from running a reverse DNS resolution against the target IP addresses.
* --system-dns, tells nMap to use the system DNS resolver instead of the predefined nMap resolver. Unlike the system resolver, the nMap resolver supports parallel call execution, therefore circumventing it can markedly increase the required scanning time, and is only used in case a bug is found.
* --dns-servers allows the user to set DNS servers for the scan manually.

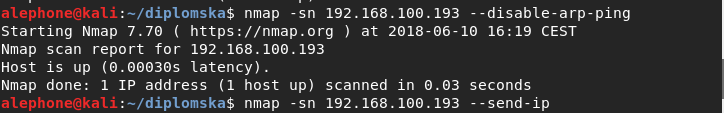
## 5.2 Port Scanning

A port scan is usually the first intrusive step undertaken in an attack. Port scanning can reveal information regarding the target’s port states, network running services as well as a more detailed network infrastructure [4]. There are several types of port scans:

* TCP SYN Scan (-sS) is the most commonly used nMap scanning technique and it’s known to be relatively fast and silent. Works by sending SYN packets, and awaits for a SYN/ACK reply if the port is open, or RST reply otherwise. If no response is received, or an ICMP Unreachable response is received, the port is marked as filtered. Any packets following the initial one in the response are discarded, so the TCP connection is never fully established which improves scanning time, and makes TCP SYN scan more difficult to detect.
* TCP Connect Scan (-sT) is a scanning technique used when TCP SYN Scan is not available, usually due to the user lacking the necessary privileges to utilize raw sockets, or the scanning being performed under Microsoft Windows, which does not offer support for raw sockets. This scanning technique uses the kernel to establish a full TCP connection with the target system, which makes it slower and easier to detect then the TCP SYN scan.
* UDP Scan (-sU) works by sending UDP packets to each target port. In some cases such as port 53 (DNS), 161/162(SNMP) the packet sent contains protocol specific data in order to increase the reply rate, however for most ports the UDP packet is empty unless specified otherwise. If an ICMP Port Unreachable Error (type 3, code 3) is received in response the port is marked as closed, while ICMP errors of type 3, and codes 0, 1, 2, 9, 10, 13 result in marking the port as filtered. If an UDP service running on the target machine sends a reply the port is marked as open. If no response in received the port is marked as open|filtered, meaning the port might be open but packet filters are blocking communication attempts. Version detection (-sV) could be utilized to differentiate open from filtered ports.
* SCTP INIT Scan (-sY) is a SCTP equivalent of the TCP SYN Scan. This scanning technique works by sending an INIT packet in order to establish a connection, and expects either an INIT-ACK reply if the port is open, or an ABORT if the port is closed. If an ICMP Unreachable Error is received, or there is not response, the port is marked as filtered.
* TCP NULL (-sN), FIN (-sF), Xmas (-sX) are scan types that exploit a loophole in the design of TCP. According to RFC 793, if the target port is closed any incoming, non-RST packet should warrant a RST response, and in the following pages RFC 793 states that if an incoming packet does not have either of the SYN, ACK or RST flags set at 1, that packet should be discarded. The NULL scan sets all flags at 0, the FIN scan sets the FIN bit at 1, while the Xmas scan sets the FIN, PSH and URG flags at . Provided the target system follows the RFC 793 recommendation, any packet that does not have it’s SYN, RST of ACK flags at 1 should warrant a RST response if the port is closed, or no response should the port be open, i.e. the packet would be discarded. These scan types can be effective is bypassing certain firewalls, and in some cases can be even more difficult to detect then the TCP SYN scan. However, many systems including Windows, a large number of Cisco devices, BSDI and IBM OS/400 do not comply to RFC 793 and respond with a RST packet regardless of the port state, which results in nMap marking all ports as closed. Another down side to these scan types is their inability to differentiate open from filtered ports because they are unable to tell if the packet as been discarded by the running service or the firewall.
* TCP ACK Scan (-sA) is a scan type used to map firewall rules and determining if the firewall is stateful. It works by sending ACK packets to the target device and listening for either a RST packet back, which would mean the port is unfiltered, or an ICMP error message which would mean the port is filtered. If no response is received the port is also marked as filtered.
* TCP Window Scan (-sW) is a scan type that is almost identical to the TCP ACK scan. It’s based on a rare implementational detail concerning the Window field in the response packets. Some systems set the field’s value at a positive number if the port is open and at zero if the port is closed. However, this is a small set of systems and the Window scan is not often reliable.
* TCP Maimon Scan (-sM) is a scan type named after it’s founder, Uriel Maimon. The Maimon scan follows a similar patters as the NULL, FIN and Xmas scan, except that with this scan the FIN and ACK flags are set to 1. According to RFC 793 this should be replied to with a RST packet, but Maimon has found that may systems, descendents of BSD simply discard the packet if the port is open.
* --scanflags allows use to craft TCP packets with our chosen flags. E.g. --scanflags URGACKPSHRSTSYNFIN crafts a packet with all flags set to 1. In addition to setting the flags, as basic scan type can be specified, such as -sA or -sF which would tell nMap about how to interpret the results. If no other type is specified, TCP SYN is used as a basic scan type.
* SCTP COOKIE ECHO (-sZ) is an SCTP scan which uses COOKIE ECHO packets and listens to ABORT reply in case the port is closed. If the port is open, by SCTP there should be no response. SCTP COOKIE ECHO scan should be more difficult to detect then the SCTP INIT scan, and can be more effective against non-stateful firewalls. As a disadvantage, this scan type cannot tell the difference between opened and filtered ports.
* TCP Idle (Zombie) Scan (-sI <zombie host>[:probe-port], otherwise knows as a blind scan, is the quietest known scan type since no packet to the target computer is directly sent from the computer performing the scan. All IP packets contain an identification field (IP ID), and with most implementations the value of this field is an integer incrementing by 1 with each packet sent. The Idle scan works by sending a (usually SYN/ACK) packet to a third party (a zombie host), expecting a RST packet in response. The reply contains the zombie host’s IP ID number. Next, the crafts a SYN packet with a source address spoofed to the zombie host’s address, and sends this packet to the target device. If the SYN packet arrives at an open port, the target sends a SYN/ACK to the zombie host in an attempt to establish a TCP connection. Since the zombie host regards the SYN/ACK packet as unsolicited it replies with a RST, and therefore increments it’s IP ID by one. If the SYN packet runs into a closed port, the target send a RST packet to the zombie host, which is disregarded and the zombie’s IP ID remains unchanged. If the SYN packet arrives at a filtered port at the target device, it’s discarded and no additional packets are exchanged between the target and the zombie, thus the zombie’s IP ID again remains unchanged. Finally, the scanner sends another SYN/ACK to the zombie and gets another RST in reply. If the reply’s IP ID’s value is larger by one then the first RST packet’s, there has been no packet exchange between the zombie and the target meaning the port is either closed or filtered. Conversely, if the second RST reply’s IP ID is incremented by two from the first one, the target’s appropriate port is open. The two advantages offered by the Idle scan are anonymity (no packet exchange between the scanner and the target), as well as the possibility that an appropriate zombie host might be able “white-listed” by the target’s firewall, and therefore the scan would yield better results. However, finding a zombie host whose IP ID field is an incrementing-by-one integer which is currently network inactive might prove difficult. Another disadvantage is the Idle scan’s inability to differentiate between closed and filtered ports.
* IP Protocol Scan (-s0) is a scan type meant to discover running protocols at the target device. This is achieved by reiteratively sending IP packets with different protocol field value in the IP header, in expectation of an ICMP Protocol Unreachable Error (type 3, code 2), marking the protocol as closed. If an ICMP Port Unreachable (type 3, code 3) is received instead, the protocol is marked as open. Other ICMP Unreachable Errors result in nMap marking the protocol as filtered. Either of these responses mean the ICMP protocol is active.
* FTP Bounce Scan (-b <FTP relay host>) is a scan type based on an old FTP specification (RFC 959), which enables establishing FTP connections by proxy. In turn, this enables users to ask an FTP server to resend a certain file to another computer. Errors received is response usually tell us whether the target port is open or closed. Although this scan type was commonly used when it was first introduced to nMap in 1997, today it’s almost never used because an appropriate FTP server is almost impossible to find.

## 5.3 No Ping Scan and appropriate IDS/IPS rules generation

We perform a No Ping Scan with ARP Ping disabled in order to force nMap to use IP packets over ethernet to 192.168.100.193 by executing nmap -sn 192.168.100.193 --disable-arp-ping (figure 6)

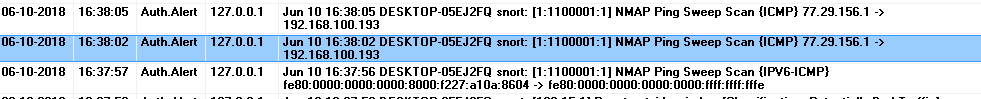
***Figure 6. No ping scan***

In Snort’s configuration we add the rule:

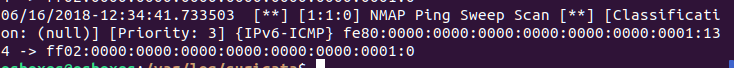
аlert icmp any any -> 192.168.100.193 any (msg:"NMAP Ping Sweep Scan"; dsize: 0; sid:1100001; rev: 1)

Accordingly in Suricata’s configuration we add:

alert icmp any any -> any any (msg:"NMAP Ping Sweep Scan"; sid:1;)

telling the IDS to alert us with a “NMAP Ping Sweep Scan” message if an ICMP packet with no data payload heading to 192.168.100.193 is caught (figures 7 and 8).

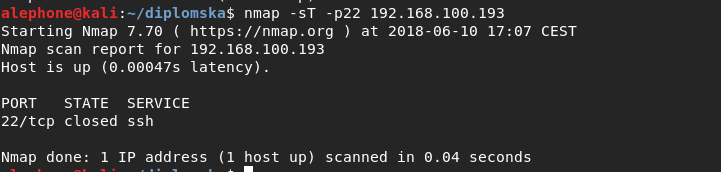
***Figure 7. Snort log of No Ping scan***

******

***Figure 8. Suricata log of No Ping scan***

## 5.4 TCP Connect Scan and appropriate IDS/IPS rule generation

Executing nmap -sT 192.168.100.193 would tell nmap to run a TCP Connect scan against 192.168.100.193 (figure 9). An attacker would oftentimes be interested in the state of port 22 (ssh) and 22 is included in the default -sT scan

***Figure 9. TCP Connect scan***

In Snort’s configuration we add the rule:

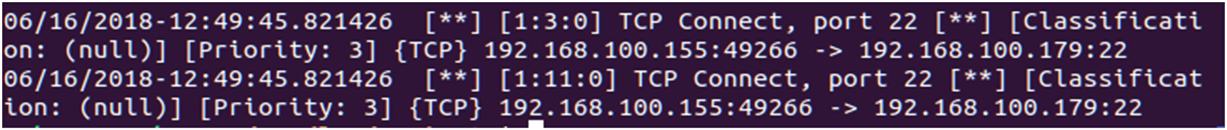
alert tcp any any -> 192.168.100.193 22 (msg:"TCP Connect, port 22"; sid:1100001; rev: 1)

Appropriately in Suricata’s configuration we add:

alert tcp any any -> any 22 (msg:"TCP Connect, port 22"; sid:3;)

telling the IDS to alert us the the message "TCP Connect, port 22" (figures 10 and 11). This rule would be of use at all addresses where a SSH connection would be inappropriate. If a SSH connection from certain addresses is appropriate, these addresses could be listed and allowed through.

***Figure 10. Snort log of TCP Connect scan***



***Figure 11. Suricata log of TCP Connect scan***

## 5.5 Xmas Scan and appropriate IDS/IPS rule generation

In Snort’s configuration we add the rules:

alert tcp any any -> any 22 (msg:"Xmas Scan, port 22"; flags: FPU; sid:1100003; rev: 1)

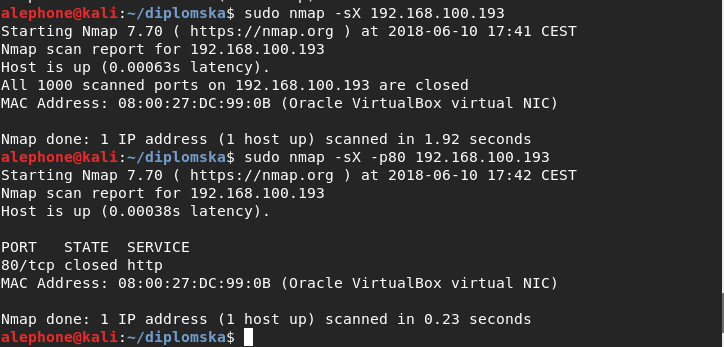
alert tcp any any -> any 80 (msg:"Xmas Scan, port 80"; flags: FPU; sid:1100004; rev: 1)

Appropriately in Suricata’s configuration we add:

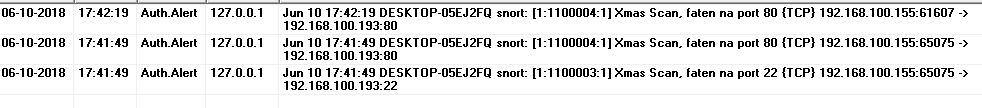
alert tcp any any -> any 22 (msg:"Xmas Scan, port 22"; flags: FPU; sid:4;)

alert tcp any any -> any 80 (msg:"Xmas Scan, port 80"; flags: FPU; sid:5;)

Now, the IDS would alert us (figures 13 and 14) if a TCP packet with it’s FIN, PSH, URG flags set at 1 and SYN, ACK, RST flags set at 0, a packet which has no actual use in a TCP connection, and only used in an Xmas scan. If these packets are discarded by the kernel, nMap would incorrectly mark all ports as closed.

In order to perform a Xmas, FIN or NULL scan nMap has to utilize raw packets, meaning the user would need root or sudo privileges.

***Figure 12. Xmas scan***



***Figure 13. Snort log of Xmas scan***

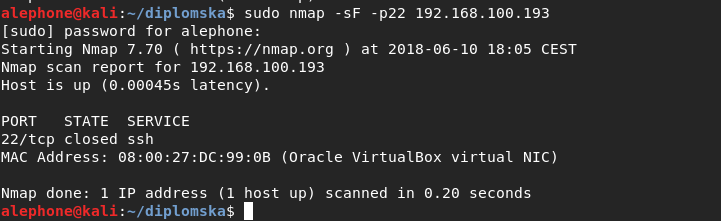


***Figure 14. Suricata log of Xmas scan***

## 5.6 FIN Scan and appropriate IDS/IPS rule generation

With FIN scanning all flag bits except the FIN one are set to zero. Using nMap we can scan port 22 at 192.168.100.193 by executing

nmap -sF -p22 192.168.100.193



***Figure 15. FIN scan***

In order to catch such a packet heading to port 22 (or any port), in Snort's configuration we add the following rule

alert tcp any any -> any 22 (msg:"FIN Scan, port 22", flags: F; sid:1100004; rev: 1)

Appropriately, in Suricata's configuration we add:

alert tcp any any -> any 22 (msg:"FIN Scan, port 22"; flags: F; sid:6;)

Using these rules, the IDS would alert us with the message "FIN Scan, port 22" each time such a packet is caught



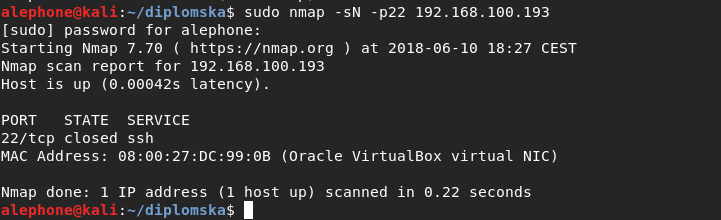
***Figure 16. Snort log of FIN scan***



***Figure 17. Suricata log of FIN scan***

## 5.7 NULL Scan and appropriate IDS/IPS rule generation

With the NULL scan, all flags (SYN, ACK, RST, URG, PSH, FIN) are set to 0. We can perform such a scan by executing sudo nmap -sN 192.168.100.193



***Figure 18. NULL scan***

Against this scan, in Snort’s configuration we add the rule:

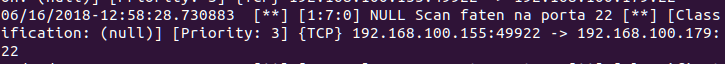
alert tcp any any -> any 22 (msg:"NULL Scan, port 22"; flags:0; sid:1100004; rev: 1)

And in Suricata’s rule set we add (figure 20):

alert tcp any any -> any 22 (msg:"NULL Scan, port 22"; flags:0; sid:7;)

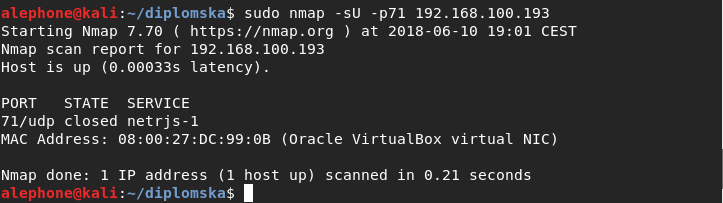
Therefore the IDS would alert us each time such a packet is discovered.

***Figure 19. Snort log of NULL scan***

 ***Figure 20. Suricata log of NULL scan***

## 5.8 UDP Scan and appropriate IDS/IPS rule generation

UDP scanning should be able to tell us whether a UDP service is running on a certain port of an active host. We could check port 71 at 192.168.100.193 by executing nmap -sU -p71 192.168.100.193 (figure 21). sudo is again required in order to be able to use raw packets

***Figure 21. UDP scan***

In Snort’s rule set we append:

alert udp any any -> any any (msg:"UDP Scan"; sid:1100005; rev: 1)

Appropriately, in Suricata’s configuration:

alert udp any any -> any 71 (msg:"UDP Scan"; sid:8;)

Such rules would alert the administrator with each UDP packet caught. This poses a problem since usually a network has a large volume of legal UDP traffic, so these rules would result in too many unnecessary alerts. This situation can be improved upon by “white-listing” addresses where UDP traffic is expected (e.g. all UDP traffic that originates from and travels to the internal network nodes), but this leaves the possibility for an attacker to abuse an otherwise trusted address to perform the scan (however the likelihood of this happening is low, as is the likelihood of an UDP scan revealing something which is not already known to an attacker that is inside the network). A different approach entails writing rules that listen to UDP traffic on ports where such a traffic is not expected.

E.g.. Snort rule listening for UDP traffic on port 71:

alert udp any any -> any 71 (msg:"UDP Scan"; sid:1100005; rev: 1)

Suricata equivalent:

alert udp any any -> any 71 (msg:"UDP Scan"; sid:8;

***Figure 22. Snort log of UDP scan***



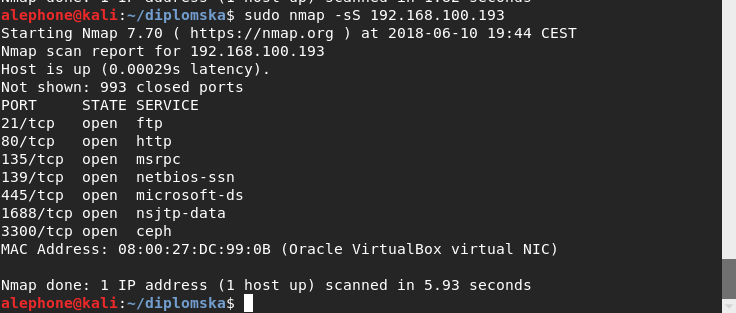
***Figure 23 Suricata log of UDP scan***

## 5.9 SYN Scan and appropriate IDS/IPS rule generation

The SYN scan, also known as the half-open scan because it never completely establishes a TCP connection to the target is the default, well-known silent scanning technique. Using nMap we perform such a scan against 192.168.100.193 by executing:

nmap -sS 192.168.100.193

This scan type requires the usage of raw packets in order to prevent the kernel from establishing a complete connection, and therefore root or sudo privileges. Instead completing the connection, nMap crafts a SYN packet, and inspects the reply (SYN/ACK, RST or none) in order to determine the port state (open, closed or filtered).



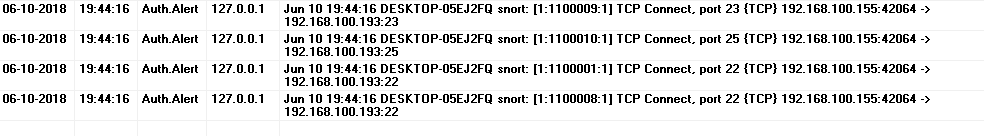
***Figure 24. SYN scan***

In Snort’s configuration we add the same rules we would add for the TCP connect scan:

alert tcp any any -> any 22 (msg:"TCP Connect, port 22"; sid:1100008; rev: 1)

alert tcp any any -> any 23 (msg:"TCP Connect, port 23"; sid:1100009; rev: 1)

alert tcp any any -> any 25 (msg:"TCP Connect, port 25"; sid:1100010; rev: 1)

Snort catches the SYN scan just like it would the TCP Connect scan:

***Figure 25. Snort log of SYN scan***

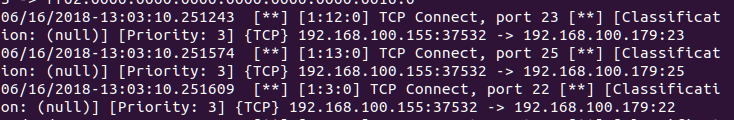
Equivalent Suricata rules:

alert tcp any any -> any 22 (msg:"TCP Connect, port 22"; sid:11;)

alert tcp any any -> any 23 (msg:"TCP Connect, port 23"; sid:12;)

alert tcp any any -> any 25 (msg:"TCP Connect, port 25"; sid:13;)

Suricata’s alerts are shown in figure 26.

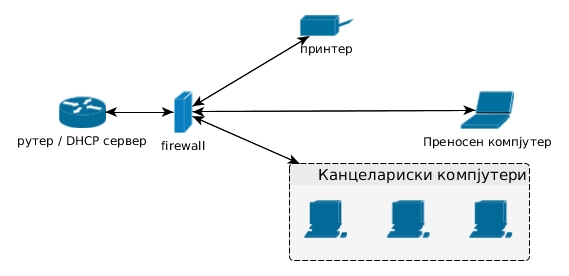
***Figure 26. Suricata log of SYN scan***

The SYN Scan can go undetected by badly configured, and usually older, stateful firewalls, however, using the listed rules Snort and Suricata mind the incoming TCP packets and are easily able to detect the SYN scan.

# 6. Creating optimal models of IDS rules concerning different network environments

## 6.1 Small and medium-sized organizations’ networks and appropriate rules

Computer networks of small and medium sized organizations are usually comprised of a single gateway, a DHCP server with an Access Control List which enables it to double as a firewall, office desktop and laptop computers, a printer and similar devices. Figure 27 depicts such a network.



***Figure 27. Network of a small organization***

In such a network we could assume a small number of functioning network protocols, as well as a small number of open ports which should be subject to monitoring. The router would also function as a DHCP server which would be an effective way to assign IP addresses to hosts [5]. DHCP uses UDP traffic at port 67 server side and port 68 client side, thus, we could monitor non-DHCP traffic on these ports. Assuming 192.168.0.1 where the router is located, and the network addresses are contained in 192.168.0.0/24, we could monitor non-UDP traffic by adding the following rules:

alert tcp any any -> 192.168.0.1 67 (msg:"TCP Connect, port 67 192.168.0.1"; sid:..;)

alert icmp any any -> 192.168.0.1 67 (msg:"ICMP packet, port 67 192.168.0.1"; sid:..;)

alert tcp any any -> any 68 (msg:"TCP connect, port 68"; sid:..;)

alert icmp any any -> any 68 (msg:"ICMP packet, port 68"; sid:..;)

ICMP Echo would be useful for running network diagnostics during deployment or maintenance, therefore we could leave port 7 open. However, we would expect it to be put to seldom use, and occasional use by an administrator aside, ICMP Echo packets might mean network enumeration in preparation of an attack. Although it might raise false positives on occasion, a rule alerting against Ping packets like

alert ip any any -> any 7 (msg:"Ping!"; sid:..;)

might help administrator oversight over small networks, and could deter many network discovery methods.

Internet Printing Protocol and CUPS (Common Unix Printing Protocol) use TCP traffic to transfer printing data, and UDP traffic to find an available printer. Both TCP and UDP streams utilize port 631. Appropriate rules here would monitor for packets of a different protocol (ICMP among others), as well as packets originating from outside the network:

alert icmp any any -> any 631 (msg:"ICMP packet, port 631"; sid:..;)

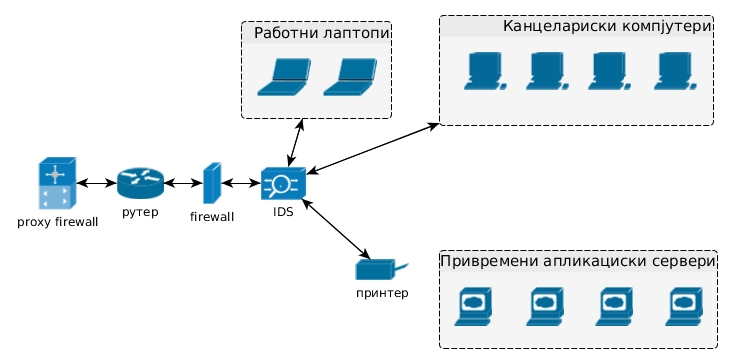
alert ip !192.168.0.1/24 → any 631 (msg:"Packet source external, port 631"; sid..;)

If a need for printing via the Internet or external network arises, a set of allowed IP addresses could be assigned to a variable ($ALLOWED\_IPS in the following rule):

alert ip !$ALLOWED\_IPS any → any 631 (msg:"Packet source not allowed, port 631"; sid..;)

## 6.2 Networks of IT organizations and appropriate rules

A dedicated firewall placed at the default gateway, workstations, application servers used in software development would be devices present in a network of an IT organization as depicted in figure 28.



***Figure 28. Network of an IT organization***

IT organizations, especially software development companies would employ a different strategy for maintaining a safe network. Local hosting of web applications during development and testing is a common practice. At this stages, application safety is not a priority which oftentimes result in unsafe code. This warrants the use of a proxy firewall, operating at the application level of the OSI model which could deter attacks like Cross Site Scripting and SQL Injections [1]. An open line of communication between the programming and the administration teams would ensure the IDS/IPS rules are configured according to the needs of the running applications. In addition, it would be expected that port 7, used for the ICMP Echo protocol is put to common use, therefore monitoring it might prove ill advised. FTP is another protocol that would be put to often use, so ports 21 and 22 (sftp) should be monitored for packets of different protocols only. For example the following rules work against ICMP and UDP packets respectively:

alert icmp any any -> any 21 (msg:"ICMP packet, port 21"; sid:..;)

alert udp any any -> any 21 (msg:"UDP packet, port 21"; sid:..;)

Port 22 could be used by SSH or sftp services, and the administrator could take steps to monitor UDP, ICMP packets, or connection attempts originating at unauthorized addresses:

alert icmp any any -> any 22 (msg:"ICMP packet, port 22"; sid:..;)

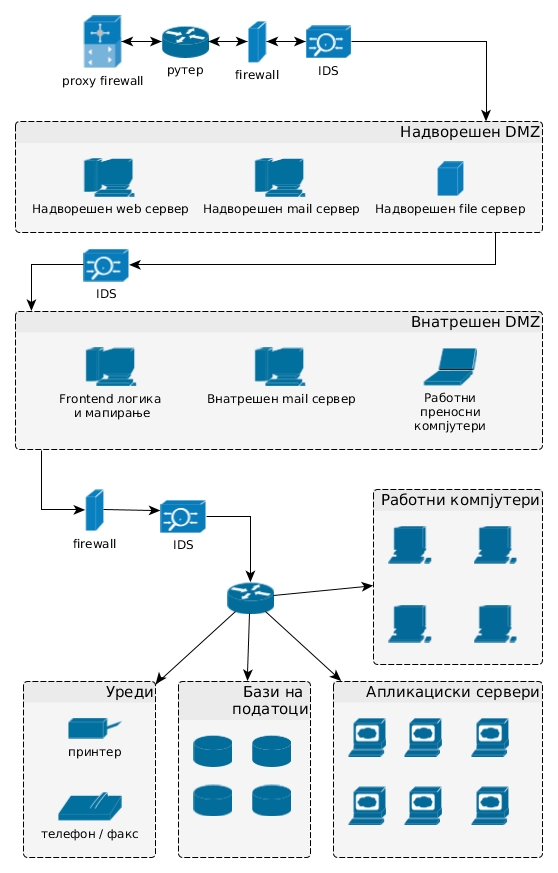
alert udp any any -> any 22 (msg:"UDP packet, port 22"; sid:..;)

alert ip !$ALLOWED\_IPS any -> any 22 (msg:"Packet from an unauthorized source address, port 22"; sid:..;)

Database management systems and the ports they use should be available to the internal network and monitored by only allowing appropriate packets, in accordance with the respective DBMS documentation.

## 6.3 Network of medium-sized and large organizations and appropriate rules

Similarly to smaller organizations, midsized and large organizations would benefit from the rules about monitoring DHCP, ICMP Echo and IPP/CUPS. It is likely that a larger organization would have network software packets running, e.g. a network of an university campus would run an e-learning software, while bookkeeping software packets and resource administration software would find wide use in many types of large organizations. Figure 29 depicts such a network topology:



***Figure 29. Network of a large organization***

Besides the mandatory presence of a firewall and the use of an IDS, additional security can be provided by dividing the network in VLANs and placing a DMZ. This would also enable the administrator to predict types of traffic and protocols used in each network segment, therefore they could adjust the IDS rules accordingly (e.g. allow communication with the bookkeeping servers only in the bookkeeping department). Unlike networks in IT companies, networks in large, non-IT organizations can be expected to have a fixed number of running services. Using the service’s respective manuals, an administrator should be able to implement IDS rules that conform exactly to the running software.

It is likely that a large organization would regularly need to transfer binary files. In university campus example, we could have an e-learning system with it’s address stored in a variable called $ELEARNING and the only people that should be able to access the e-learning system via FTP are the teachers, If we store the teachers address in a variable called $TEACHERS we could utilize a rule to alert the administrator of a connection attempt from an address not belonging to a teacher as follows:

alert ip !$TEACHERS any → $ELEARNING 21(msg:”Unauthorized FTP”; sid:..;)

In a similar manner, if the bookkeepers whose addresses are stored in a variable $BOOKKEEPING are the only ones allowed to send POST requests to the bookkeeping software located at an address stored in $BKKP\_SW:

alert ip !$BOOKKEEPING any → $BKKP\_SW 80(msg:”Unauthorized POST ”; content:”POST”; nocase; sid:..;)

would alert the administrator of any POST requests made to the bookkeeping software from a unauthorized address.

If the organization with addresses stored in $ORG uses an internal DNS, with an address stored in a variable $DNS, the administrator could monitor for packets with external origin with the rule:

alert ip !$ORG any → $DNS any (msg:”Foreign packet”; sid:..;)

As well as monitor for packets sent from anywhere to a port other then 53 on the DNS:

alert ip any any → $DNS !53(msg:”Packet at wrong port”; sid:..;)

If an organization has it’s own mail server, monitoring it’s port 25 (SMTP) and 110 (POP3) would be advisable. Provided the mail server’s address is stored in the variable $MAIL, the following rules would alert the administrator of ICMP or UDP packets sent to the open ports:

alert udp any any → $MAIL 25(msg:”UDP packet, port 25”; sid:..;)

alert icmp any any → $MAIL 25(msg:”ICMP packet, port 25”; sid:..;)

alert udp any any → $MAIL 110(msg:”UDP packet, port 110”; sid:..;)

alert icmp any any → $MAIL 110(msg:”ICMP packet, port 110”; sid:..;)

Principles listed above would not be applicable to telecommunication networks which use PPP and other protocols, but could still be used in network segments that work over IP and standard protocol stacks. Hosting companies would need to append a series of more specific rules, according to their infrastructure. The emergence of the ever more popular cloud architectures, significantly altered the security scene. From the cloud user perspective, the amount of work required to keep the network safe is decreases as the number of services and applications that are hosted by a cloud increases. However, from the perspective of cloud operators, the security of their implementation becomes a much more dynamic problem.

# 7. Conclusion

Intrusion detection systems provide the administrator of a certain network with an additional chance to prevent an attack in it’s beginnings, as well as to quickly mitigate the damage caused by an already executed attack. From an organizational perspective, this could prevent losses in capital, workflow and trust acquired. The functionality of such a system depends on the quality of it’s configuration and administration. IDSs based on rule sets usually come with a large number of predefined rules, and it is common practice for these rules to be left active on the network regardless of the fact they are not applicable to it due to the limited number of running services. This leads to unnecessarily checking network packets against patterns which are unusable in the network, which, in turn, leads to devices overuse, large number of false positives, wrongfully discarding packets or incomplete traffic analysis. Earlier in this paper we looked at generic cases, and it became apparent that a preliminary idea about the applications and services that would run on the network can be formed. Later, the administrator could use this idea to predict the network traffic and write appropriate rules which would significantly improve the IDS’s efficiency. We have also looked at rule set models in three generic types of networks, upon which can be further worked on in order to make them applicable in a specific network environment.

Although there has been a small amount of research concerning IDS usage in a cloud environment, it has been shown that standard rules apply there as well. Cloud specific rules is a topic that is in it’s infancy still. We could hypothesize that a development in cloud specific rules would include automatic rule generation with user creation, as well as bringing about of changes in privileges in a way that the IDS would function in a similar manner as an identity server, however, independent of user applications and the security mechanisms of the cloud platform.

# 8. References

[1] Glen E. Clark, CompTIA Security+ Study Guide, Third Edition, McGraw-Hill Education, 2018

[2] Sean-Philip Oriyano, CEH v9 - Certified Ethical Hacker Version 9 Study Guide – 3E, Third Edition, Sybex, 2016

[3]  Shahid Anwar, Jasni Mohamad Zain , Mohamad Fadli Zolkipli , Zakira Inayat, Suleman Khan, Bokolo Anthony and Victor Chang, From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions, Multidisciplinary Digital Publishing Institute, 2016

[4] Peter Kim, The Hacker Playbook 3 Practical Guide To Penetration Testing, Independently published, 2018

[5] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach, Sixth Edition, Pearson, 2012

[6] Gordon Fyodor Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning, Nmap Project, 2009