Anna Drozdova

T561SN

SECURING A DNS SERVER WITH SNORT IDS

Severen-Telecom case

Bachelor's Thesis
Degree programme of Information Technology

October 2015

MAMK
University of Applied Sciences

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis** 09.10.2015
Author(s) Anna Drozdova	**Degree programme and option** Information Technology

Name of the bachelor's thesis

Securing the DNS server with Snort IDS: Severen-Telecom case

Abstract

Today Severen-Telecom is one of the largest Internet providers for corporate clients in the Saint-Petersburg region. When analysing log files of IDS, it was noticed that the company's DNS server from time to time received queries to resolve non-existent domain names of Chinese web-stores. These requests come from the client hosts.

The purpose of this study was to investigate the architectures of Intrusion Detection System and Domain Name System. Another aim was to find out the way to protect the DNS server. In order to reach the goal of the study the IDS architecture was studied and Snort IDS was chosen. The Snort system is one of the most powerful IDSs today. The possibility to work in the inline mode allows not only detecting suspicious activity, but also blocking it. Manually written Snort rules are a helpful addition.

All examinations were performed in a virtual laboratory environment and were not tested in real conditions. And as a result it was discovered that Snort IDS was not able to block all fake DNS queries alone, only partial filtering. This study is not finished at this point; complete blocking of packets generated by the bot may be performed with different Snort plug-ins of additional software.

Subject headings, (keywords)

IDS, Snort IDS, rules, DNS, inline mode

Pages	**Language**	**URN**
62	English	

Remarks, notes on appendices

Tutor	**Employer of the bachelor's thesis**
Matti Juutilainen	Mikkeli University of Applied Sciences, Severen-Telecom

CONTENTS

1 INTRODUCTION

Over the past two decades, information technologies have made a real breakthrough. The advent of hypertext, IP-telephony, increase in CPU processing speed, bandwidth of communications, and much more, had appeared. All these innovations significantly complicated the process of the maintenance of IT infrastructure. Today the network technologies are present in all the spheres of human activity. The question of creating its own local network arises in almost every company at some point. Together with this, another question should come up, too - the protection of confidential information, because the advent of new technologies is closely connected with the advent of different vulnerabilities that may harm the end users as well as the entire network. Therefore, network security should be an important part of the administration and its maintenance.

The main causes of all the threats and attacks are the disadvantages of modern information technologies and the constant growth in complexity of software and hardware. Also openness, the scale and heterogeneity of networks significantly increase the level of vulnerability of computers and their data.

The Domain Name System is one of the most important parts in the Internet space and one of the most attractive ones for attackers. And the Severen-Telecom, one of the largest Internet providers in the Saint-Petersburg, Russia, is not an exception. The company's DNS server is being attacked periodically. And the aim of this study is to show how the company's DNS server can be protected. For this purpose Snort Intrusion Detection System was chosen.

The structure of the work is the following: the first four chapters are the theoretical part. Chapter 2 has overview of network security. The next paragraph describes the Domain Name System. Chapters 4 and 5 tell about Intrusion Detection System and, especially, about the chosen system – Snort. The rest of the thesis work is the practical implementation of Snort IDS in a laboratory environment in order to protect DNS server of Severen-Telecom company.

2 NETWORK SECURITY

Network security is a sum of all actions taken to prevent any loss. Securing requires a wide variety of different security measures and this process is continuous — it is not enough to apply only those measures which were known at the time of installing servers and configuring the network. Security administrators must stay aware of all new technologies and security issues and apply new security measures constantly [1].

In order to properly protect the network, it should be known what could be damaged, what threats and security methods exist. Also, when creating secure network it is very important to follow the standard model of a computer network based on three main concepts:

1. Confidentiality. Information should be accessible only to authorized persons.
2. Integrity. Information should be protected from unauthorized changes.
3. Availability. Sustainable access to information should be provided to users with access rights [7], [8].

Based on these concepts of security, the key principles of network security can be determined as follows:

1. Making security easy: simplifying the network and breaking it into small parts.
2. Recording all attempts to accesses the data.
3. Giving the right users an unlimited access to the right information.
4. Isolating resources as much as possible.
5. Managing security risks: learning the network, its vulnerabilities, types of threats and protection methods that can even prevent attacks [5].

2.1 Classification of unauthorized access methods to data

The methods of unlawful access to protected information can be divided into several sections by different principles. FIGURE 1 presents this classification [7]:

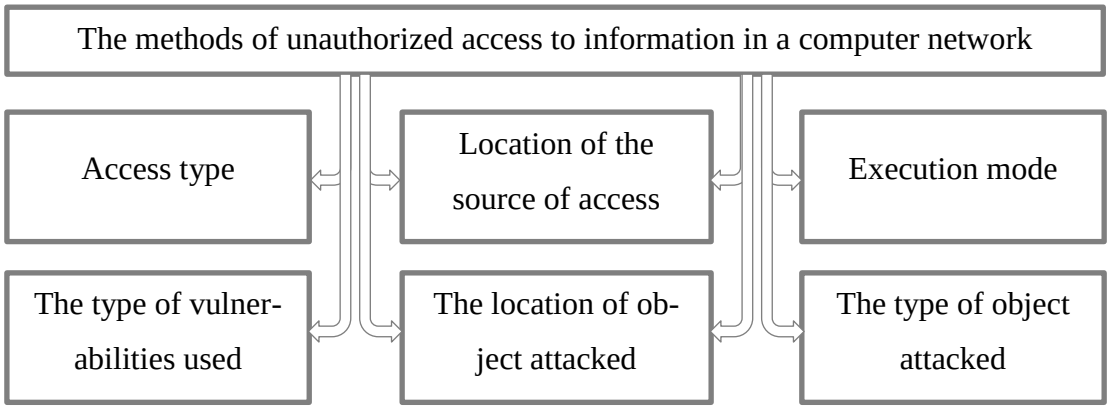


FIGURE 1. Classification of unauthorized access methods.

The types of access methods shown above should be reviewed closely:

1. The access type attacks can be physical and logical. Physical access can be implemented by several methods: by stealing storage media, through visual interception of information, eavesdropping and interception of electromagnetic rays. Logical access, in turn, involves the logical overcoming of the active protection system of the network. This type of unauthorized access is the most effective for the intruders [7].
2. The location of the access attacks are divided into two categories: attacks, the source of which is located in the local network and attacks, the source of which is located outside the local network. In the first case, very often the initiator of the attack is the user with access rights. The second category includes attacks on private networks which have open access to networks such as the Internet [7], [8].
3. The execution mode attacks can be carried out at constant human intervention and carried out by specially developed software which is based on virus technology [7].
4. The type of vulnerabilities and weaknesses used attacks can be the following:
    a. Attacks based on the deficiencies of security policies. The presence of these errors means that the security policy developed does not reflect the real aspects of the information use.
    b. Attacks based on the errors of administration. This refers to incorrect or insufficient maintenance.
    c. Attacks based on the deficiencies of the security algorithms. The use of imperfect encryption algorithms gives effective results in attacks.

   d. Attacks based on the errors in the implementation of the security system [7], [9].

5. By the location of the attacked object attacks can be:
   a. Attacks on the information stored on an external storage device.
   b. Attacks on the information transmitted through the communication lines.
   c. Attacks on the information processed by the computer [7].
6. By the object attacks are divided as follows:
   a. Attacks on the security policy and administrative management process.
   b. Attacks on the permanent/removable components of the protection system.
   c. Attacks on communications protocols.
   d. Attacks on the elements of a computer system.

   The object of attack is the object, analyses or use of which allows successful access to protected information. This classification is the most important and enables more accurate methods to differentiate attacks, to analyze and determine the necessary protective measures [7], [8], [9].

2.2 Types of attacks

Today a huge number of different attacks can be performed, and a part of them can be regarded as the most common ones. And all these attacks can be divided into several big groups according to their vector. The attack vector of an attack is the main means by which the attack reaches its target. Identifying this attack vector allows to accurately describe attacks and provide better security, because the behaviour of the attack is its most important feature [9].

The following classification presents the division by vector definition. By using this categorization each existing attack belongs to one of the nine categories [9]:

1. Virus. The type of malware propagates by inserting a copy of itself into some program. Through travelling from one computer to another it leaves infected files. Almost all viruses need to be with an executable file. This means that a virus can be present on a computer, but is not active until the moment when

the malicious program is opened or executed. And, it spreads to other hosts only with this executable file, using e-mails, network, disks or file sharing [6].

2. Worm. This is self-replicating program which works without infected executable files. It uses networks or e-mails to send copies of itself and to infect new hosts. The main difference from viruses is that worms are standalone programs, which do not need human help to propagate [6], [9], [10].
3. Trojan. This malware is named after a wooden horse the Greeks used to attack Troy. It is a harmful piece of software that looks legitimate. Unlike worms and viruses, Trojans can't self-replicate and spread through user interaction (opening an e-mail attachment, downloading and running a file from the Internet). After it is activated, it can achieve any attacks on the computer, from irritating the user (ex. changing desktops) to damaging the host (deleting files, stealing data, or activating other malware). Very often Trojans are used to create backdoors to give malicious users access to the system [6], [9], [11].
4. Buffer overflow. This type of network attack occurs when the program writes more information into the buffer than the space it has allocated. This allows an attacker to overwrite data that executes the program and to establish the control of the program in order to execute the malicious code. Very often buffer overflow is used as a background to implement more serious attacks [9], [15].
5. Denial of service (DoS). It is an attack which makes special resources unavailable to users and intended hosts. These victims' resources can be network bandwidth, services, data and others. Distributed DoS attacks take place when many compromised devices act and flood the victim simultaneously and are under control of a single attacker. An attacker brings vulnerable hosts under control using software exploitations (like buffer overflows). Special rootkits can also be used in host systems to create weaknesses and backdoors in the network. In the worst cases, the victim system can be permanently damaged. [9], [16], [17], [18], [19].
6. Network attacks. These attacks focus on networks and their users. This is the largest category of attacks, including manipulating different network protocols on all possible layers – from the data-link layer to the application layer. Most common attacks in this category are the following: different manipulations with wireless networks, spoofing (attacks where a person or a program mas-

querades as another by falsifying data) and web-application attacks (attacks on databases, on temporary files and cross site scripting) [9], [20], [21], [22].

7. Physical attacks. These attacks focus on damaging physical network components and computers. For example, an attacker can use different types of energy weapons: high-energy or low-energy radio frequencies to garble stored or transmitted data. A famous attack in this category is Van Eck phreaking. It is a form of eavesdropping which works with special equipment, and data can be captured from keyboards, computer displays, printers, and other electronic devices by their electromagnetic fields [7], [9], [23].
8. Password attack. This type of attack is focused on users, on their credentials in order to steal sensitive information. Three main methods can be used in this case: Brute Force, Dictionary attack and Sniffing (by using the Key logger program). The first two methods are just guessing the password, while the third one is more complex, and because of this the probability of a successful attack is higher [9], [24].
9. Information gathering attack. This type of attack implies further actions by attackers. At this point there is no damage to the information or network components. Data is only collected for analysis and further attacks [7], [8], [9].

Summary of all the attacks is present in TABLE 1.

TABLE 1. Vector's categorization

Level 1	Level 2	Level 3
Viruses	File infectors System/boot record infectors Macro	
Worms	Mass mailing Network aware	
Trojans	Backdoor Exploit Rootkit	
Buffer overflows	Stack Heap	
Denial of service	Host-based	Resource hogs Crashers
	Network-based	TCP flooding UDP flooding ICMP flooding
	Distributed	

Network attacks:	Spoofing	
	Session hijacking	
	Wireless attacks:	WEP cracking
	Web application attacks:	Cross site scripting Parameter tampering Cookie poisoning Database attacks Hidden field manipulation
Physical attacks:	Basic	
	Energy weapon:	HERF LERF EMP
	Van Eck	
Password attacks:	Guessing:	Brute force Dictionary
	Exploiting implementation	
Information gathering attacks:	Sniffing:	Packet sniffing
	Mapping	
	Security scanning	

2.3 Types of security assessments

To be sure that a network is secured, there are different types of security assessments. Knowing them and to be able to use them are necessary measures to verify the level of security of the network resources. These assessments are the following [8]:

Vulnerability Scanning. This is the basic type of security assessment and helps to discover widely known weaknesses. Frequently, it requires administrative rights and special software, but also scanning can be performed by custom scripts. Most scanning software perform actions like enumerating computers, OSs and applications, identifying security mistakes, testing for exposure to simple attacks and searching for computers with vulnerabilities. This method does not require a lot of skills to conduct [1].

Penetration test. This is a more complex type of security assessment which scans the network as a whole, not only individual computers and network components. This testing can discover how vulnerabilities are exploited and what weaknesses related to people present in the network. But, this requires high IT skills [1].

IT Security Auditing. This type is very complex and time-consuming set of different actions, and it assesses the effectiveness of security policies and procedures. Auditing

will help to determine whether an organization has necessary components for securing the network. Often, IT security auditing is performed by the third organization which is specialized in network examinations [1].

3 DOMAIN NAME SERVERS

The Domain name server (DNS) is a distributed dynamic database that contains information about the hosts in the Internet. Distributed means that each server does not contains information about all of the Internet namespace. Each server is responsible for its sector of the space. Most commonly, it includes the machine name, IP address and the data for mail routing [8], [28].

This system was introduced for the users' convenience and the main task is the conversion of human-readable computer names into IP addresses and vice versa. Therefore, in a case when a computer has lack of information about certain device, it sends a request to the DNS server to obtain the necessary data [27].

3.1 DNS name hierarchy

The domain structure of DNS is a tree hierarchy consisting of nodes, areas, domains, subdomains, and other elements. The top of the domain structure is the root zone. Setting the root zone is located on multiple servers/mirrors placed around the world. This contains information about all servers in the root zone, and is also responsible for the top level domains (ru, net, org, etc.). The servers of the root zone process and respond to requests, giving information only about the top level domains. The root and top level domains are administered by The Internet Corporation for Assigned Names and Numbers (ICANN). Below the top level domains (TLD), the administration of namespace is delegated to organizations [27], [29].

FIGURE 2 shows the DNS hierarchy tree. Each node in the tree represents a DNS name. This name can include DNS domains, computers or services. DNS domains can contain both hosts (computers or services) and other domains (subdomains). Each organization is assigned authority for a part of the domain namespace and is responsi-

ble for administering DNS domains and computers within this sector of the namespace [27], [30].

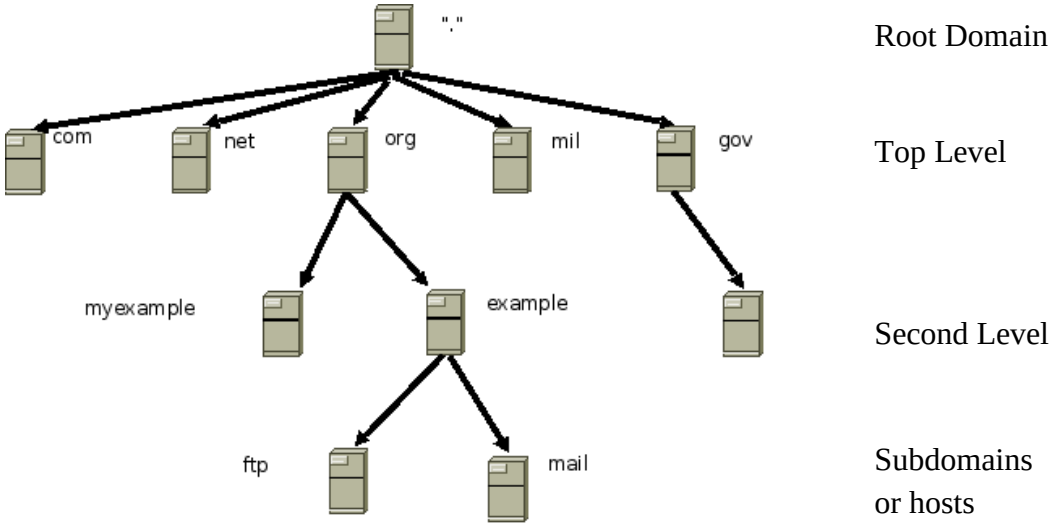


FIGURE 2. DNS hierarchy

Zone - is any part of the tree of the domain name that is placed as a whole on a DNS server. Zone can be called a "zone of responsibility". The purpose of the allocation of the tree in separate areas is the transfer of responsibility to persons or organizations. Each zone has at least one authoritative server DNS which stores all the information about the zone for which it is responsible [28], [30], [32].

Domain - is a named branch or subtree in the tree of DNS structure, therefore, it is a particular node, including all subordinate nodes. Each node in the DNS hierarchy is separated from its parent by a dot. The domain name begins with a dot (the root domain) and passes through the domains of the first, second, and third (if necessary) levels and ends with a hostname. Thus, domain name fully reflects the structure of DNS hierarchy. Often, the last dot (the designation of the root domain) in the domain name is omitted. For example, in the browser, it is used not google.com., but google.com [27], [28], [30].

Fully Qualified Domain Name (FQDN) - is the domain name that uniquely identifies the domain name and includes the names of all parent domains in the DNS hierarchy, including the root. It is like an analogue of the absolute path in the file system. An

example of this is present on FIGURE 3 that introduces the domain name ftp.example.org from FIGURE 2[28], [30]:

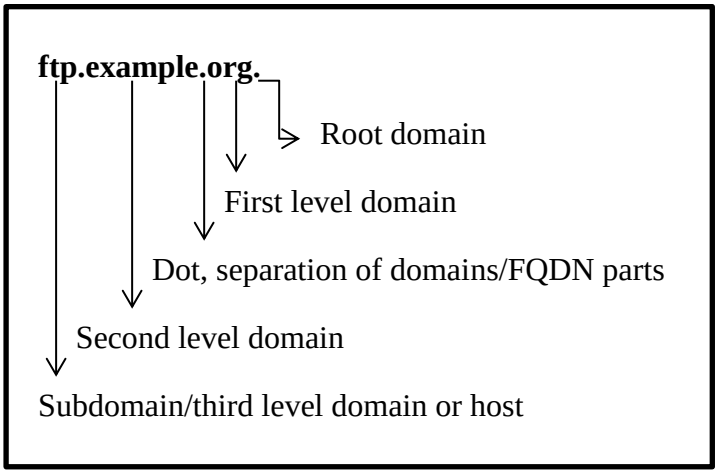


FIGURE 3. Fully Qualified Domain Name example

3.2 Resource records

This is actually for what DNS exists. Resource record is the unit of storing and transmitting information in the Domain Name System. Each entry contains information of the names and related service information in the DNS, for example in the pair domain name - IP address [28], [35].

The resource record consists of the following fields:

NAME. *TTL* *CLASS* *TYPE* *DATA*

1. NAME - domain name to which this record/IP address belongs to. In the absence of this field, the resource record is inherited from the previous record.
2. Time To Live (TTL) – lifetime of the entries in the cache DNS, after this time the record is deleted. This field can be not specified for individual resource records, but then it must be stated at the beginning of the zone file and will be inherited by all records.
3. CLASS - defines the type of network. 99.99% of the cases used IN, which means the Internet. This field has been created with the assumption that the DNS can work in other types of networks than TCP/ P networks.

4. TYPE - type of record and purpose of the record.
5. DATA - various information. Its format and syntax are defined by TYPE [28], [35].

The most used types of resource records are the following:

1. A (address record) represents the host name (domain name) of the IPv4 address. For each network interface of the machine there should be made one A record. For example, the following entry displays the domain name example.com. to an IPv4 host address 73.167.234.56 (field NAME – example.com., field TTL - 86400, the field CLASS - IN, the field DATA - 73.167.234.56):
   example.com. 86400 IN A 73.167.234.56
2. AAAA – IPv6 address record. It is similar to the A-record.
3. CNAME (canonical name record) displays the alias to the real name (to redirect to a different name). For example, the following entry specifies the alias ftp to host www.example.com.:
   ftp 86400 IN CNAME www.example.com.
4. NS (name server) points to the DNS server for the given domain. If NS record refers to the name server for the current zone, the domain name system does not use them practically. IT just explains how the zone is organized and which machines play a key role in providing the service.
5. MX (mail exchange) specifies hosts to deliver mail addressed to the domain. The NAME field specifies the destination domain. Fields TTL and CLASS have the standard values; the TYPE field is set to MX. The DATA field specifies the priority and the domain name of the host responsible for receiving mail. For example, the following entry indicates that the mail for example.com should be routed on mx1.example.com. And if mx1.example.com has problems mail should be routed to mx2.example.com. Moreover, for both MX hosts there must be relevant A-records:
   example.com 86400 IN MX 10 mx1.example.com
   example.com 86400 IN MX 20 mx2.example.com
6. SOA (Start of Authority) describes the initial zone configurations and defines the area of responsibility of the server. For each zone there should be only one SOA record, and it should be on the first position, because the SOA record applies only to those records that follow below it.

7. PTR (Pointer) displays the reverse mapping from addresses to names (inverse A record) [28], [34], [35].

3.3 DNS query message

As FIGURE 4 shows, each message starts with a header which contains the *identification* field that allows connecting requests and responses in pairs. The *Flags* field determines the nature of the request as well as the response. Four other fields determine the number of entries of a certain type contained in the message. For example, *number of questions* specifies the number of entries in the section *questions* that require answers. Each question consists of the domain name, followed by the type and class of the request [27], [30].

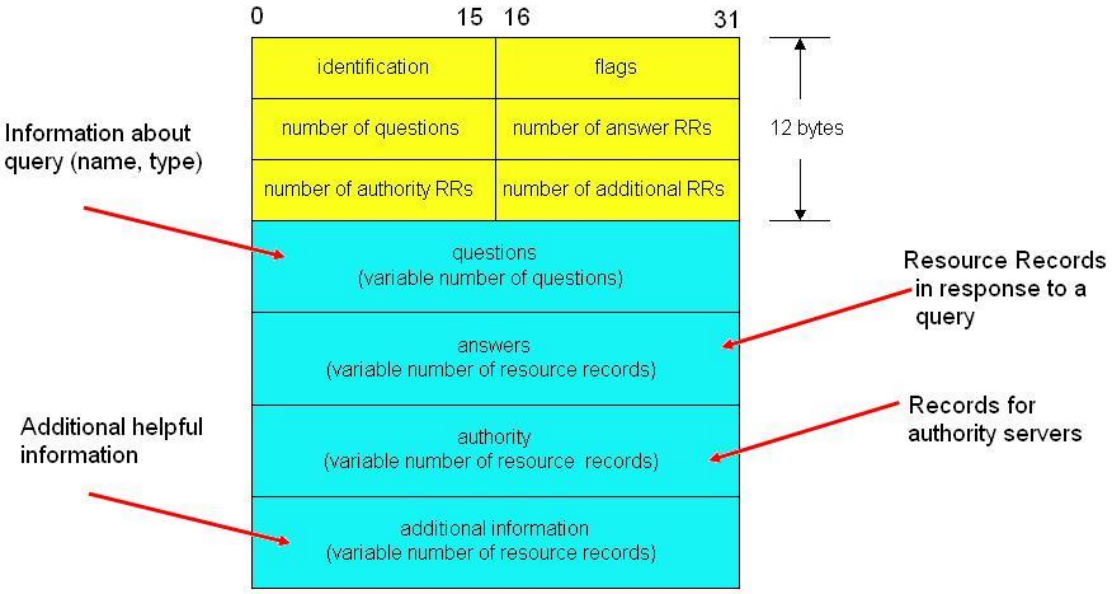


FIGURE 4. DNS request message format

The most important part of the message shown on FIGURE 4 is the *questions* section that contains the main information of the packet – request for resolving the domain name. There are three fields here, and they are *domain name*, *request type* and *request class*:

1. The first field (*domain name*) has a variable length, including one or more subfields. Each subfield begins with a byte of its length, and the field ends with 0.

For example, for the domain name www.example.com the field is 3www7example3com0.

2. The next field determines the type of the request. Its values are the same as in the type field of resource records. For example, the A query requires the IP address resolution, the MX query asks about IP addresses for mail delivery, and so on.
3. The request class field usually has a value of 1 indicating Internet addresses [27], [30].

3.4 How does DNS work?

There are two ways to resolve names. The first one is recursive - a method in which the DNS servers do all work to give final data back to the host. The second is an iterative query: the server will give back a referral to other DNS servers which might have the answer [28], [31].

The recursive server returns only real answers and error messages. It keeps track of references that are exempt from this duty of the client. The basic procedure for the resolution of the request, in essence, is the same. The only difference is that the name server takes care of the processing of references, not passing them back to the client [28], [31].

This method works as follows:

1. The resolver of the client sends a recursive query to a DNS server on some Server1.
2. Server1 has no needed data (it is not authoritative for the required zone or there are no records about it in the cache) and sends a query to the Root.
3. Since the data cannot be resolved (root works only as an iterative server), the root server returns to the Server1 IP address of the DNS server which is responsible for the .com zone.
4. Server1 sends a query to the server responsible for the .com zone.
5. Since data cannot be resolved, the server returns to the Server1 IP address of the DNS server responsible for the area example.com

6. Server1 sends an iterative query to the server responsible for the example.com area.
7. Server1 gets the desired data (for example, mail.example.com).
8. Server1 sends the data back to the client's resolver [28], [31], [33].

In iterative query if the server has an address, cached from a previous request, or if it is authoritative for the domain, it will give an appropriate response. Otherwise, instead of the correct answer, it gives a reference to the authoritative server of another domain that should know the answer. In this case, the job of finding the answer lies in the local client's resolver. This method takes more time to get the answer [28], [31].

3.5 DNS response message

DNS responses can be authoritative and non-authoritative. Authoritative answers come from servers which are responsible for the zone, while the second type of answers comes from servers that have needed records in their caches and which are not responsible for the zone [27], [28].

The DNS responses typically contain the following information:

1. Record title - information about the request.
2. Record Request - repeats the request.
3. Record response - actually, the answer.
4. Entries about authoritative servers - information about authoritative servers that store information about the current request.
5. Further information - additional records, such as NS server addresses [27].

3.6 Weaknesses of Domain Name System

DNS is one of the frequently attacked elements in the network. In terms of security there are two main causes of malfunction:

1. Remote attacks on the DNS server, namely:
   - An introduction of an intermediate host between the attacked object and the server or substitution (correction) of the zone information.
   - An attack on the DNS response by falsifying DNS servers.

2. Administrators' mistakes: incorrect designation of the correspondence between the IP address and host name [27], [30].

Today there is no perfect protection against these attacks, but a set of extensions can be used to minimize it. This system is called the Domain Name System Security Extensions (DNSSEC). Its main features are as follows:

1. DNSSEC is digitally signed. Thus, DNS client can verify the loyalty and integrity of information.
2. In addition, this system can verify cryptographic certificates of general purpose, allowing using DNSSEC as a global distributed storage of signature key certificates.
3. But DNSSEC does not provide confidentiality of data. In particular, all DNSSEC answers are authenticated, but not encrypted.
4. DNSSEC does not protect against DoS attacks by itself, although in some ways does this indirectly [27], [36].

4 INTRUSION DETECTION SYSTEMS

Intrusion Detection System is software or hardware that captures network and host activity data in event logs and provides tools to generate alarms and query and reporting tools to help administrators to analyse data during and after the incident. Different IDSs can perform a variety of functions, but no one IDS does all of these:

- Monitoring the system and users' activity
- Recognizing known attacks in the system
- Analyzing system configurations for misconfigurations and vulnerabilities
- Identifying suspicious activity through statistical analysis
- Correcting errors is system configuration
- Making traps to record data about intruders [3], [4].

As a result, this is just one element in the security architecture. It doesn't provide any protection measures, only analyzing tools. Like a security camera in a building doesn't eliminate the need of door locks [3].

4.1 Elements of an IDS

Every Intrusion Detection System consists of four main blocks introduced in FIGURE 5 [3], [4]:

<table>
<tr><td rowspan="3">Management:
Configuration, Tuning</td><td>Action:
Alarms, Queries, Reports</td></tr>
<tr><td>Analysis:
Attack Signatures and Anomaly Detections</td></tr>
<tr><td>Logging (Data Collection):
Individual Events are Time-Stamped
Log is Flat File of Events</td></tr>
</table>

FIGURE 5. Elements of IDS

Then, these four main blocks can be described in more detail as follows:

Logging (Data collection).

This is the first step of the working process. This element of IDS captures the discrete activities, such as the arrival of the packet or attempt to log on. Each activity is time-stamped and saved in a sequential file sorted by time. These log files are an essential part of the system, because all analyses are performed based on this data [3], [4].

Analysis.

After this, all collected data should be analyzed. To do this, two methods exist and based on it there are two different types of IDSs, too:

1. Attack Signatures. This is the simplest IDS analysis method similar to static packet filtering.
2. Anomaly Detection (Heuristics). Analysis method in which heuristic patterns try to detect variations from historically normal operations [3], [8].

Actions.

The third element of the system is actions which are generated automatically or with human interaction based on analyzed log files. IDS can perform a few different actions in this case:

1. Alarms should be as special as possible to give the administrator comprehensive information about the problem and what can be done with it.
2. Retrospective interactive data analysis helps the administrator to make sense of the collected data by providing interactive query tools. By using these tools administrators can better understand thousands of entries in log files and the security problems of network.
3. Periodic reporting shows a summary of suspicious activity and a broader picture of the network for some period of time.
4. Automated actions don't require any human participation. These limited actions are based on previous event patterns [3], [4], [8].

Management.

The fourth element and its settings are the basis for the operation of the entire system. In other words, all IDSs are needed to be correctly configured, updated and managed all the time in order to work properly. Poor configuration makes IDS useless [3].

4.2 Distributed IDSs

Distributed IDSs provide more complex and more useful system. When all the four elements presented in FIGURE 5 are on one device this IDS is limited in use. Often, it is necessary to analyze traffic which goes through multiple hosts to see the broader picture of network incident. Therefore, distributed IDSs are used in these situations. This means that one central manager console collects and analyses data from several agents (FIGURE 6) [2], [3], [4].

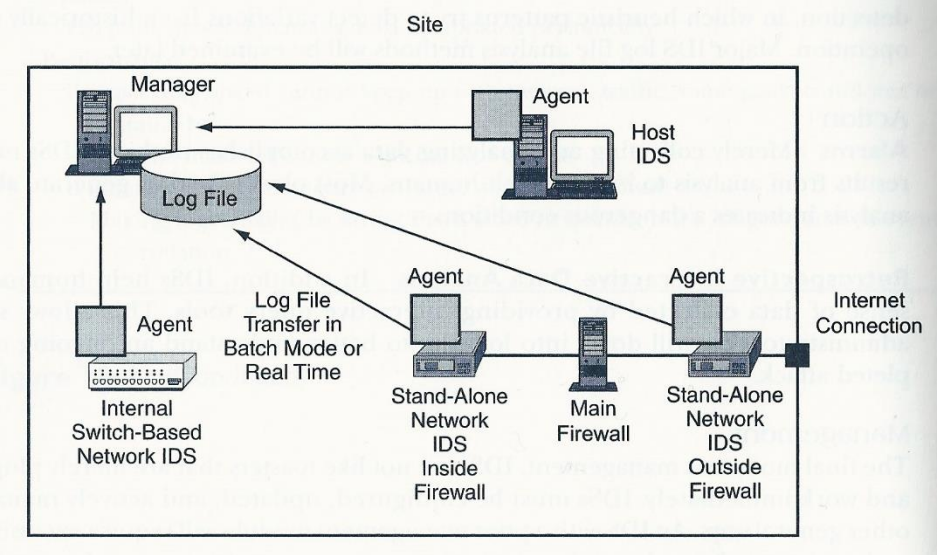


FIGURE 6. Distributed IDS

Agents. This is the software on the monitoring device (host) that collects data and can sometimes do simple analysis and alarm reporting. In cases of manager's heavy work its simple analysis can help a lot, but on the other hand, all actions are based on the small piece of collected data (by one agent). Therefore, it can be incorrect, and false positives (false alarms) can be generated [2], [3], [8].

Manager (Central Manager Console). This element of IDS is responsible for collecting all log files from agents into single log file, analysing its data, and generating alarms and intelligent queries. The manager exists on a separate device. To provide security, such IDSs should use authentication and encryption to make communications between the manager and agents trusted [2], [3], [8].

Data Transfers. Collected data can be transferred from agents to the manager in two different ways: through batch mode and real-time mode. The batch-mode transfer is more general and less CPU consuming, and minimizes the load on the network. In this mode agents collect data for some time - several minutes or hours - and then send blocks of log files to the central manager. During the real-time mode, each event's data is sent to the manager immediately. It is not widely used mode, because it interrupts the manager often [2], [3], [4].

4.3 Types of IDSs

Different goals require different things to use. The same is with IDSs. Some of their specific types are described in the following sections [3].

4.3.1 Network IDSs.

Network IDSs (NIDSs) capture packets as they transfer through networks. FIGURE 6 shows an example of a stand-alone Network IDS. Elements are plugged into the network and work in a promiscuous mode, so it captures and analyses all packets that are passed by them. They are like corporate sniffers. Another type of NIDSs are router and switch IDSs which are built into the routers and switches, respectively and scan by default all ports [3], [4], [8].

The placement of NIDSs can vary in the following ways:

1. Between the Border Firewall and the Internet. This placement allows NIDS to listen the all coming traffic. Therefore, it is the best way of placement in many cases.
2. Between the Border Firewall and the Internal Network. This type of placement helps to ease the overload of log files. Because of the Border Firewall, not all traffic can pass and NIDS will only listen to packets which are allowed by the firewall.
3. Inside the Local Network. Placing NIDS near the sensitive and important servers will strengthen the security level and protect the network from inside attacks by dishonest employees or compromised computers [3], [8].

The main benefit of using NIDSs is that such agents detect all packets which are passing through some location. NIDSs have the high level of diagnosing the attacks. But on the other hand, there are some important drawbacks. Firstly, using only the border NIDSs generates big blind spots in the internal network, because routers and switches are not able to see packets. And another weakness is encrypted data. Firewalls are not able to see all packet data, only the unencrypted part (ex, IP addresses). Also, NIDSs cannot scan such packets and when most part of the data transferred through the network is encrypted, NIDSs become less effective [2], [3].

4.3.2 Host-based IDSs.

Such IDSs work with data collected based on a host computer and allow monitoring connections, programs and activities of a specific machine. HIDSs perform like a host/personal firewall and only help to protect that one host. These IDSs can be integrated into a centralized IDS system. HIDSs recognize when the system operates in the normal state and in case of some problem react with alarm or action [2], [3].

Host IDSs can work in three different ways: Protocol Stack monitoring, Operating System monitoring and Application monitoring. In the first case HIDS monitors packets coming into and going out of the host. It is like with NIDS, but only in the scope of a single device. Sometimes it is also possible to see packets after decryption. Another way to execute HIDSs is monitoring OS events. A lot of useful information can be collected by such IDSs, for example failed logins, users accessing critical or unusual files, creating new accounts, modifying executables or registry keys and changing or deleting files. The third type of HIDSs works on the application layer. Although it is an uncommon way to work, these logs may help to explain the meaning of logs at two other levels. Furthermore, some application-specific attacks can be recognized with these IDSs [3], [8].

Host IDSs are a useful and powerful supplement to the Network IDSs, but despite this there are two huge weaknesses:

1. Host IDSs have a limited picture of what is happening in the network. They only see events on a specific host.
2. Host IDS is executed on one host and therefore the host is attractive to attackers. It can be compromised and all log files can be deleted or changed [3], [8].

4.3.3 Signature-based or pattern-based IDSs

This method of detection attack attempts is based on the analysis of the forwarded packet, and in comparison, it results with the rule base containing signatures of all known malicious network traffic instances. So, alarms or actions are generated only if the traffic matches some database record. The signature database should always be kept up to date. Otherwise, it will be ineffective, but many pattern-based IDSs are

flexible enough and their databases can be modified according to custom rules, for example, based on needs of certain network [2], [4], [8].

Statistical analysis is used in signature-based IDSs. This means that a single instance of event is not a problem, but many instances of the same event in a short period of time can indicate a problem. As a result, frequency is the basic measurement value in the statistical analysis [2], [3].

4.3.4 Heuristic or anomaly-based IDSs

Instead of looking for matches in the database (as in pattern-based IDSs), heuristic IDS is looking for behaviour that is not normal for the network or individual host. IDS focuses on individual characteristics of the packets to see if they comply with the standard of normal activity. That is why, it is easy to determine with the anomaly-based IDS that the system is compromised [2], [4].

An alternative for such IDSs is the Misuse Intrusion Detection System. It works based on the model of known bad activity. This means that all real actions are compared against known suspicious ones [4].

4.4 Limitations of IDSs

Intrusion Detection Systems are relatively new products on the IT market. They were introduced only in mid 1990s. Therefore, IDSs are constantly evolving and improving, allowing administrators to protect the network from more and more sophisticated and dangerous attacks. IDSs are not perfect systems and a big problem of all types is sensitivity. It is very hard to find the point of balance between false alarms and missed attacks. One of the main problems with signature-based IDSs is a delay between the moment when a new problem is detected and the moment when its signature can be added into the system [4], [26].

Finally, these systems are not program independent. They require continuous human participation and are a nice addition to the security system. IDSs should be monitored and configured to work properly. Otherwise they will be useless [4].

5 SNORT

Nowadays the most powerful Network Intrusion Detection System is Snort. It is a free and open source program under the GPL license. Snort was created and became available in 1998 by Martin Roesch - one of the most famous person in the world of Information Security - and was developed and managed by Sourcefire. The first version was only a Linux-based packet sniffer. In 1999 Snort was already able to make rule-based analysis. Since 2013 this company and the rights for Snort have belonged to Cisco Corporation. According to its author, the main reason for Snort creation was the absence of sufficiently effective and free tools for alerting about vulnerabilities and attacks on the networks. Today, Snort is the most spread IDS in the world due to its openness and hard work of authors [2], [12], [13], [39].

Snort is also a well-functional packet sniffer and packet logger. In addition to these features, Snort supports other useful functions such as sending real-time alerts, closer examination and the filtering of entries in log files and live traffic sampling. And, it can be used as an Intrusion Prevention System (IPS). To work as an IPS it is enough to set up Snort in the inline mode and then passing traffic will not be only analyzed and logged, but also can be blocked. This feature is often used with possible DDoS attacks. [12], [37], [38].

The Snort system can be considered as lightweight network-based IDS, because it has a relatively small footprint and it does not require a specialized server to work. An important advantage of this IDS is the opportunity to work on different operating systems, both on open source OSes and on commercial ones. For example, Snort works with different UNIX platforms (eg. OpenBSD, Solaris), Mas OS and Windows OS. [12], [39].

Snort is a signature-based IDS, using rules to check for suspicious packets. A rule is a collection of requirements based on which an alert is generated. The size of these rules is increasing constantly as the number of known exploits is also increasing. Now this rule library is organized by the categories of attack and vulnerabilities and updated regularly. These rules can be downloaded from the official website at snort.org [12], [41].

Snort should be updated regularly to keep it up-to-date with the latest bug patches and the latest collections of rules. This is because like every application Snort may be subject to attacks, for example, DoS attack [37], [38].

With the help of the Snort system a lot of different attacks can be revealed, such as the following:

1. Attacks on Telnet, FTP, DNS, etc
2. DoS/DDoS attacks
3. Attacks associated with Web servers (CGI, PHP, FrontPage, ISS, etc.)
4. Attacks on the databases (SQL, Oracle, etc.)
5. Attacks on protocols SNMP, NetBios, ICMP
6. Attacks on SMTP, IMAP, POP2, POP3 [12], [13], [14].

Snort also allows detecting some viruses, bad traffic, the use of exploits (Shellcode), various backdoors, and it can perform a scan of the system (such as OS, ports, etc.) [12], [13], [37].

Snort supports interfaces with different data transfer protocols to listen to the traffic. These include:

1. Ethernet
2. SLIP (Serial Line Internet Protocol)
3. PPP (Point-to-Point Protocol) [37], [41].

5.1 Snort architecture

Snort consists of four main components which can be customized with different plug-ins (FIGURE 7):

1. The sniffer. This component allows eavesdropping on network traffic. Most often it is IP traffic, but it may also be other transfer protocols. For example, it can be used to perform a network analysis or troubleshooting. As Snort is a packet-based system, within this component it can save packets and process them later.
2. The preprocessor. At this point the packets captured are statistically analysed and sorted by checking for a certain type of behaviour. When it is determined,

a packet is sent to the detection engine. On this level many different plug-ins can be used to achieve better performance.

3. The detection engine. When packets have been handled by the preprocessor, it is time to perform the main action – the check the packet's data against the rules. If there is a match between the data and the rule, the packet is sent to the alert processor. Otherwise it just goes out. This component is the most important and largest part of the Snort system.
4. The output (alerting/logging). In the case of a match at the detection engine, an alert is generated. It can be stored in log files/database, sent by email, and other different ways may be used. This depends on the enabled output plug-ins [13], [39], [41].

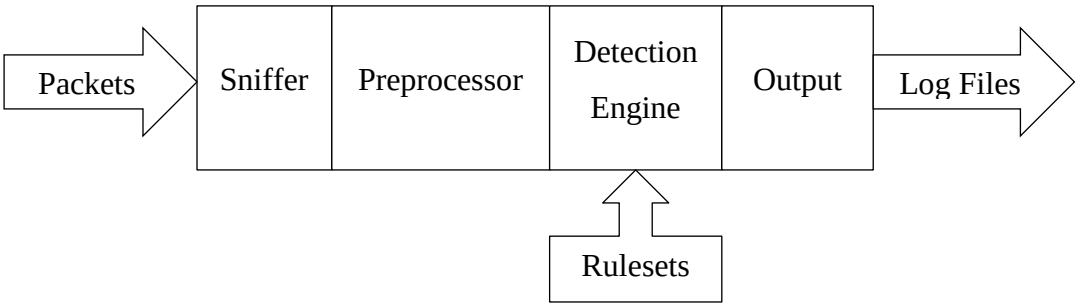


FIGURE 7. Snort architecture

The most basic mode of work is the packet sniffer. But by using other components these packets can be processed and checked against the set of signatures [13], [38].

5.2 Working modes

Depending on the intended purpose of the system use, Snort can be started in three major modes which include:

- Packet sniffer
- Packet logger
- NIDS [13], [39].

Sniffer mode. This mode is the easiest, the program only reads packets and displays information about them constantly in the console window (Terminal). There are several different options to display [37], [40].

snort –v This command only shows TCP/IP headers.

snort –vd Instructs Snort to show all Network layer headers (TCP, UDP, ICMP).

snort –vde TCP/IP headers, packet data and Data Link layer headers will be shown.

When writing commands there is no difference in which order attributes are written. And, the line *snort –v –d –e* gives the same output as the previous one [13], [37].

Packet Logger Mode. This mode performs the logging of the packets to the disk. Also, there are several options to configure the system correctly. For example, it is possible to assume which network is the home network or to tell Snort to log packets in a binary mode. With this option packets will be logged in tcpdump format [13], [39].

snort –dev –l ./log	Tell system where to store logs.
snort –dev –l ./log –h 192.168.1.0/24	Assume home network.
snort –l ./log –b	Perform binary logging [13], [39].

Binary format is more convenient for the Snort, because packet collection is, in this case, performed much faster as there is no need to translate data into human-readable format. In order to read the log files the next command should be used and the last option performs filtering, if it is needed:
Snort [-d|e] –r ./log [tcp|udp|icmp] [13], [38], [39].

To make advanced filtering Berkeley Packet Filter (BPF) may help. It allows using several options such as ignoring traffic to or from a certain host/network/port. The syntax is: *snort –vd –r ./log [filter options]* [13], [37].

Network Intrusion Detection System Mode. This mode is the most complex, but most configurable. It works with the *snort.conf* file, where all the rules and configuration options are mentioned. Based on this configuration file Snort performs detection and further analyses of suspicious traffic.
snort –dev –l ./log –h 192.168.1.0/24 –c snort.conf [13], [37].

For users this mode offers the opportunity to write their own rules. This is an extremely convenient feature and makes the system very flexible. As new vulnerabilities are constantly emerging and very suddenly, their own rules allow users to customize the

system to take into account all the weaknesses of the network, and not wait for the new version of Snort or service pack with the necessary rules [13], [38].

5.3 Snort rules

In the detection engine all data is checked against the set of rules in order to find suspicious activity. Understanding these rules and its syntax is essential and critical for the administrator to make the future work of Snort, future alerts and log files meaningful and useful [12], [13], [39].

At this moment there are about 9,500 different rules for Snort. All of them are in four primary repositories:

1. Snort.org GPL rule set. This is the original repository and its rules are of high quality, well documented and free to use. This series is a “must have” for every Snort installation.
2. Vulnerability Response Team (VRT) rule set. This ruleset is under commercial license, but five days after the release it is free to download. Rules are also of high-quality and well-tested.
3. Bleeding Edge Threads rule set. It is a collection of rules which are not maintained by Sourcefire, as the other three sets. Generally rules are of good quality, but very special.
4. Community rule set. These are light tested rules and sometimes of lower quality than the three others. Otherwise, this series is useful [13], [37].

The Snort rule is a combination of two parts: rule header and rule option. The example below is a Snort rule from the *icmp.rules* file [42]:

alert icmp $EXTERNAL_NET any -> $HOME_NET any *(msg:"ICMP ISS Pinger"; itype:8; content:"ISSPNGRQ"; depth:32; reference:arachnids,158; classtype:attempted-recon; sid:465; rev:3;)*

As the example above showed, the part in bold is a rule header portion and this is structured strictly (everything before the parenthesis). The first element is an action (*alert*), and there can also be other options such as *pass*, *drop*. The next element is

protocol (*icmp*), and it can also be *TCP, UDP* or *IP* (includes all three). Also other protocols can be specified and used. *$EXTERNAL_NET any -> $HOME_NET any* portion means that the rule is for all packets that go from the external network from all ports to the home network to all ports. Here, instead of typing each time IP addresses variables are used [13], [37], [39].

The last portion in the parenthesis is a rule option (rule body). It is not structured as the header part. The first option in the example above is *msg*, the message or rule title. If the packet matches this rule, *ICMP ISS Pinger* will be in log file to describe the problem. *itype* checks for a specific icmp type value. *content:"ISSPNGRQ"; depth:32;* – This is what should be found in packets within the first 32 bytes, not including the packet header, only the payload part. Reference option (*reference:arachnids,158*) allows quick references to the external attack identification systems. The next element (classtype) is used to categorize rules. The last two elements work in a pair. Together *sid* and *rev* form a unique identifier that each rule must have [13], [37], [39].

5.4 Data analysis

An important part of intrusion detection is data analysis. A huge number of different logs and alerts can be created, but without analysis all this information is useless and the network may be protected incorrectly [37], [40].

To perform data analysis there are four steps in advance:

1. Real-time alerts. To communicate with the administrator Snort uses these messages when the detection engine finds the match between a rule and a packet.
2. Attack detection. Snort is able to detect activity that is not an attack itself, but with the analysis of large amounts of data in log files it can be seen that such actions are systematic and are a part of a bigger attack.
3. Incident analysis. When a real attack is detected it is time to answer the questions like “what happened”, “when it happened”, “how it impacted the system”, “how it happened” and some others.
4. Reporting. In order to make improvements it is essential to report all analysis, all answers and all, even small, findings. Reports may also include predictions about future problems and attacks [12], [37], [38].

To have the bigger picture, deeper analysis of all possible data sources should be used. It can be data from firewalls' alerts, authentication logs and DNS logs, and so on. Also, a good way to have additional information is to use application log files and vulnerability scanners. All these data should be used to build a more complete picture of suspicious activity [12], [13], [37].

As it was mentioned above, Snort can be used with different plug-ins. Some of them can help at this level. Today a variety of free and commercial tools is available and most of them provide a friendly graphical user interface [12], [37], [41].

5.5 IDS Snort and DNS

One of the Snort features is the ability to protect the Domain Name System from attackers. There are two different possibilities which can be used together or not. The first one is checking packets at the preprocessor level. In the current version of Snort there is a special DNS preprocessor which is disabled by default. With correct settings this plug-in checks the following exploits: DNS Client RData Overflow, Obsolete Record Types and Experimental Recort Types, and it looks the packets over UDP and TCP. The second possibility is rules. At the level of detection engines some other problems can be found. They are such rules as "DNS name version attempt", "DNS named authors attempt" and "DNS zone transfer attempt". All the rules that can be applied to DNS traffic are collected in one file called *dns.rules* [13], [39], [42].

6 CURRENT SITUATION AT SEVEREN-TELECOM

Today Severen-Telecom is one of the largest suppliers of Internet services and telephony for corporate clients in the Saint-Petersburg region. The company network now is working properly, but there are small prerequisites that problems can occur in future.

There is a company network that includes all office workers and clients (in different VLANS and subnets). By using different monitoring tools and analysing network traffic for a long period it became possible to observe the suspicious activity of some hosts in the network. Periodically, the DNS server of the company receives packets that require resolving fake third-level domain names of the Chinese online-shops. These shops are real, but in their domain zones there are no such subdomains. These requests are very rare and do not load the network and the DNS server. Thus, the company's Department of Security does not worry about it.

6.1 Data sources

The main source of information is Snort IDS log files. Snort keeps track of all traffic n the company. Different output plug-ins are helpful here, because they give a better understanding of the company's security system. In Severen-Telecom's case Sguil [43] was used for visualization. It is a set of plug-ins with a simple and clear interface. In case of an attack, the program gives the administrator full details needed to learn the problem and make the right decision. All recorded information can be extracted and analysed using prebuilt SQL-queries. Such queries may be formulated manually and saved for later use, if necessary. Sensors are also used in addition to Snort: Barnyard collects data from Snort log files and stores them in real time in a database, and SANCP, Security Analyst Network Connection Profiler, collects, records and analyses statistical information on TCP/IP sessions. The main difference of Sguil from other projects, whose task is to deliver information collected by Snort, is the output of data in real time and the opportunity to carry out the necessary analysis to study a particular event.

In addition to Snort information from other sources was used, such as NFSen [44] and Cacti [45]. The first, NFSen, is a graphical web-based tool for analysing netflow data.

A huge variety of graphs and reports based on them allowing analysing in detail information for long periods of time. To monitor the activity of our bot, the system proved to be extremely useful. The second tool Cacti, also web-based, provides a huge range of graphic templates based on data logging and graphing system called RRDTool.

6.2 Goal of the work

The objective of my work is the development and testing of the system to protect DNS server in the lab environment. This is due to the fact that there is no high activity of bots. But this work will not remain only in the lab, it aims at full employment in the company network and future protection of the DNS server.

Currently the DNS server does not have any special protection system, just the firewall at the entrance to the local network, and there is a distributed IDS which is not configured to block requests to the DNS server. The project provides installation of the IDS client at the key position - in close proximity to the DNS server. This is necessary for better collection of necessary information about the traffic and for its subsequent analysis. Due to the fact that the work involves the subsequent introduction into the real network of the company, in order to achieve this goal it was modelled as an “ideal” system in which a client of a botnet poisons DNS servers regularly.

7 INSTALLATIONS AND CONFIGURATIONS

To create the protection system, debugging and testing of the whole scheme server is allocated for virtualization. And commonly used VMware ESXi 5.1.0 is used for the purposes of this study.

7.1 Experimental laboratory

At this hypervisor there are three virtual machines with installed GentooOS. Three is enough to achieve the goal: the first machine is infected and attacks, the second one is the victim, which must be protected, and the third one should be configured to filter

requests to protect the victim. The whole scheme of the established network is presented in FIGURE 8:

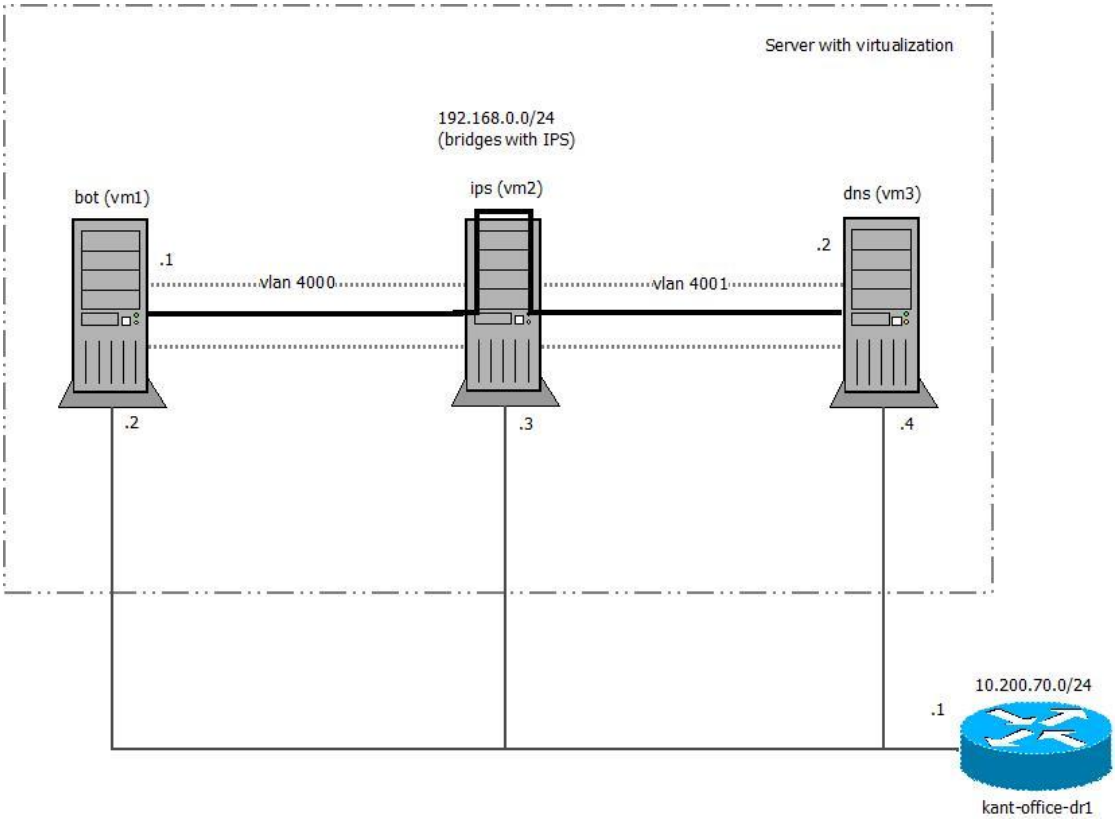


FIGURE 8. Experimental laboratory architecture

There is no need for access to the Internet, so ports that are connected to the router *kant-office-dr1*, are turned off and not used.

Machines are on the same network 192.168.0.0/24, as in the real network of the company, but for ease of configuration are located in different VLANs, bot is in VLAN 4000 and DNS is in VLAN 4001. Accordingly, one of the IPS interfaces is a member of VLAN 4000, and another of VLAN 4001.

7.1.1 DNS virtual machine

This is a machine to be protected. This host gets a huge number of requests from the bot, and, as a result, failure can occur. The goal of this work is correctly written rules for the Snort machine, so there is no need to install and configure any DNS functions. Attention is only on the IPS machine configurations. Just packet caption utility is

needed, and for this purpose tcpdump is used. Tcpdump is a UNIX utility that allows capturing and analysing network traffic going through the host.

The victim machine has the following configurations:
CPU: 1 vCPU
Memory: 256 Mb
Hard Disk: 10 GB
Operating System: GentooOS
Tcpdump: packet caption utility

7.1.2 Bot virtual machine

This virtual machine is an attacker and has in its configuration a specially crafted script that sends to the DNS server a certain number of requests per minute. It is "infected machines" from which we must protect the server.

The attacker machine has the following configurations:
CPU: 1 vCPU
Memory: 256 Mb
Hard Disk: 10 GB
Operating System: GentooOS
Script: sends packets

7.1.3 IPS virtual machine

The virtual machine that works as a bridge between the other two is IPS. It has no configured ports and all packets are going through it. There is Snort IDS installed which monitors all traffic, requests from the bot. The machine is called Snort IPS because it is not only looking for packages and notifying the administrator, but also runs on written rules that allow blocking the attempts to attack.

The Snort machine has the following configurations:
CPU: 1 vCPU
Memory: 512 Mb

Hard Disk: 20 GB
Operating System: GentooOS
Snort: intrusion detection system (in the inline mode)

More resources are allocated for this machine because the handling of data requires more memory to be as fast as possible.

7.2 Installations on IPS machine

This is the most important machine in the company's architecture. It has the Snort IDS program configured in the inline mode. Due to the working mode Snort receives more functions and becomes an intrusion prevention system.

For the program installation archive was downloaded from the official website snort.org, extracted and installed with correct settings. All commands are performed under root privileges. Without it some actions are not allowed and configuration may be incomplete:

wget https://snort.org/downloads/snort/snort-2.9.7.5.tar.gz
tar xvfz snort-2.9.7.5.tar.gz
cd snort-2.9.7.5
./configure --enable-sourcefire
make
make install

There is no need to download and install rules, because it is just a laboratory environment and the attacker can only send DNS requests. And, in the real environment basic set of rules must be installed as follows:

wget https://snort.org/rules/community
tar -xvfz community.tar.gz -C /etc/snort/rules

In the collection of rules there are special rules for the DNS server, but they are general in nature and prevent from common problems. All these rules work with two

types of network – home and external. Moreover, all of them only generate an alert about problems found. And, in this case, the attacker and the victim are in one home network and Snort should prohibit traffic passing. Thus, there is no need to download and install it.

Snort installation requires a few configuration points. All the needed configuration settings are in one file *snort.conf*. Most often basic variables should be configured, such as the addresses of the home network and external network and other different settings that are important for correct work. As Snort will work as a bridge between two hosts in one network, there is no need to specify home and external networks. Only inline mode settings should be mentioned necessarily:

39 *config daq: afpacket*
40 *config daq_dir: /usr/local/lib/daq*
41 *config daq_mode: inline*
42 *config daq_var: buffer_size_mb=256*
68 *config policy_mode: inline*
268 *output alert_fast: stdout*

Modified *snort.conf* file with bolded changed settings can be found in Appendix A. Most unneeded and commented content was deleted from file. As Snort will work in the inline mode, there is no need to download and install any other programs. Now, Snort is installed and ready to work.

7.3 Installations on the Attacker machine

For the bot machine there is no special utility, just a script that sends packets to the victim machine. This script is manually written on the C language with the help of the Internet and C language manuals. Request packets are sent by groups. The number of packets per second depends on the variables' configuration.

The executable script is in Appendix B. For the lab environment and for ease the domain name of company - severen.net - was used instead of the Chinese web-shop

names. Therefore, the scrip generates requests to the third-level domain names of the company’s namespace.

8 ATTACK SIMULATIONS AND ANALYSIS

To solve the problem - prevention of traffic from the bot to the DNS server - I have studied all the nuances of writing rules for the Snort system, all the syntax of the rules. As a result of these analyses I am able to obtain the following:

1. Snort rules do not allow sorting the traffic on "good" and "bad" and block only fake queries for the certain domain name generated by a bot.
2. It is possible to block only a part of the requests to the DNS server, for example, when the number of requests for a certain period of time exceeds the limit.
3. As the clients send its requests from different ports (not 53th port), filtering by port is not possible.

The difficulty of this issue lies in the fact that the infected hosts, in addition to the randomly generated queries to the DNS servers, send real requests; hosts are used for working purposes. One more important aspect is the fact that the shops actually exist and have real domain namespace. Users of the infected hosts may intentionally send requests to the DNS servers to resolve the existing domain names of the web-shops.

In connection with these conclusions two versions of events should be considered:

1. Complete blocking of the queries to the DNS server from the infected hosts to resolve the domain names of the Chinese shops (in the case of the laboratory environment - the domain name of the company).
2. Filtering the queries to the DNS server from the infected host to resolve the domain names of the Chinese shops (in the case of the laboratory environment - the domain name of the company). In this case a part of the packages will pass and load the DNS server, and another part will be blocked by Snort.

The case in which nothing is blocked is not considered, since at the moment the situation is exactly the same. The hosts send requests to the DNS server and Snort records

this activity in the log files and generates alerts without blocking packages. Each option is considered individually.

8.1 The first case: blocking

This scenario assumes the full block of requests to the DNS server from the infected hosts to resolve a specific domain. In the laboratory environment this domain is severen.net. In a real network it is several domains of the Chinese web-shops. In this regard, the rules are now written for only one domain. For the present network of the company this scenario requires writing several rules for each domain name. For virtual network it is enough to use the next rule to secure the DNS server:

*drop udp any any -> any 53 (msg:"BOT IS ACTIVE"; *
*content:"severen"; nocase; *
gid:1; sid:10000001; rev:1;)

As the laboratory network includes only three hosts, there is no need to write rules for specific IP addresses, but for the real network the IP addresses of the infected hosts and DNS server should be added. In other words, this rule will drop all packets going from all the IP addresses and from all the ports to the port 53 - the dedicated port for Domain Name System - in case that the packet's content matches the rule *content* option.

The *content* option gives Snort instructions about what should be found in the packets. The *nocase* option allows eliminating the difference between the lowercase and the uppercase. The combination of *sid* and *rev* assigns a unique identifier to the rule. The option *gid* allows grouping the rules. Snort rule syntax allows writing rules not just on one line, and, for the convenience, transfer rule parts on the next line using the backslash.

Since the communication between clients and the DNS server goes through the UDP protocol, Snort is configured to analyze UDP packets. But, it is also possible to communicate through the TCP protocol, if the UDP packet exceeds 512 bytes. In the laboratory, there is no need to add a rule for the TCP protocol, since the bot script does

not generate packets larger than 512-byte. But in the real network, a rule for the TCP protocol must be added as follows:

*drop tcp any any -> any 53 (msg:"BOT IS ACTIVE"; *
*content:"severen"; nocase; *
gid:1; sid:10000003; rev:1;)

None of the three machines have a graphical user interface, and all actions are performed through the console.

Now, it is time to test the new rule:

1. Tcpdump utility is configured on the virtual machine, replacing the DNS server. To do this, the next command is entered:
   tcpdump -n -i enp2s2
   -n displays the IP address instead of the hostname
   -i enp2s2 indicates which interface is listened to
2. Next Snort is configured with the command:
   snort -c /etc/snort/snort.conf -i enp2s1:enp2s2 -Q
   -c /etc/snort/snort.conf shows the path to the configuration file
   -i enp2s1: enp2s2 indicates the range of interfaces that
   -Q starts Snort in inline mode

3. After all preparations, the script is running to generate packets with the command:
   ./botlike

As the result of this experiment all sent packets are blocked by Snort. It is shown in FIGURE 9 and in FIGURE 10. There are no request packets captured by the tcpdump utility, but there are records on the Snort console. The number of records is ten and the number of generated requests is also ten. As a result, everything is well and all the generated packets are dropped by the Snort system.

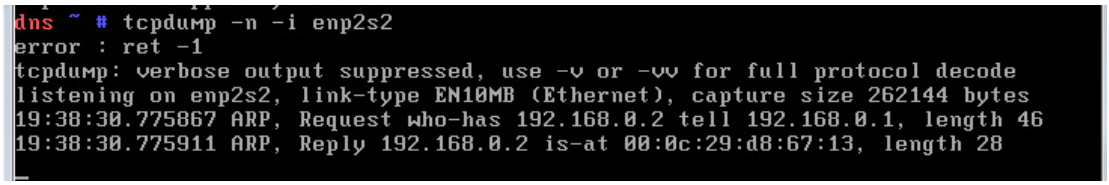


FIGURE 9. Tcpdump output, the first scenario

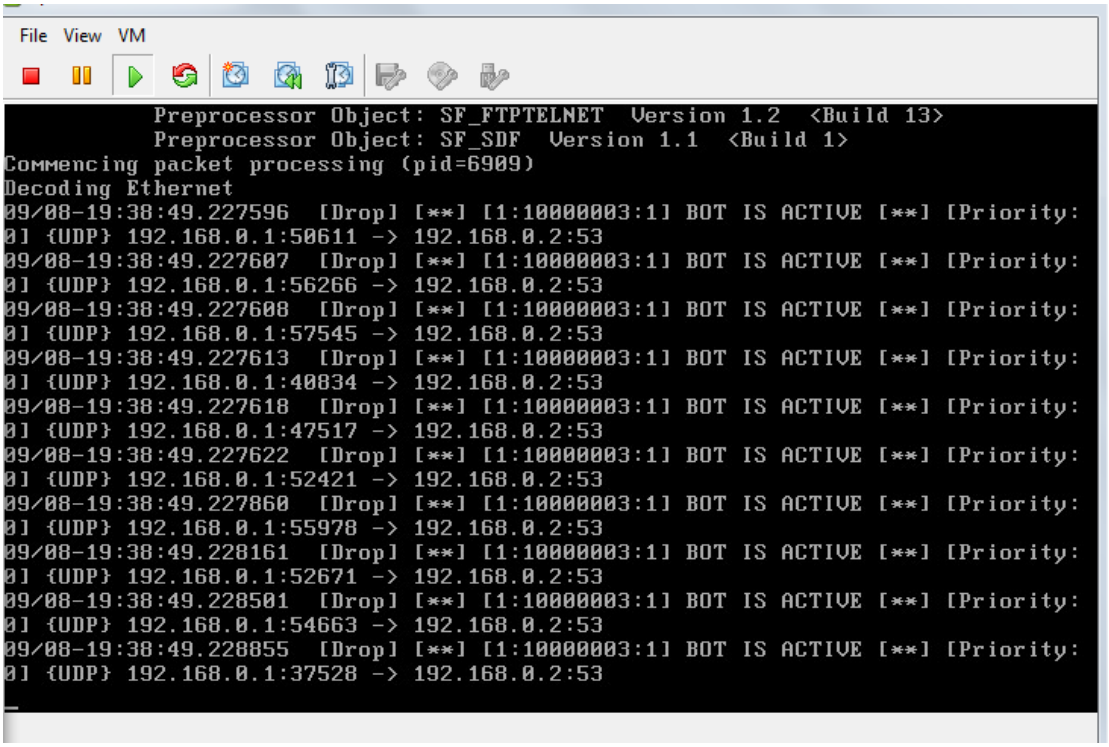


FIGURE 10. Snort IDS output, the first scenario

The tcpdump utility captures only ARP (Address Resolution Protocol) packets, because it is allowed by Snort. While Snort alerts have all the needed information about the blocked packets: IP addresses and ports (source and destination), protocol used, actions performed by Snort and a unique identifier of the rule that generates these alerts. In real conditions this scenario will block all the packets that contain requests for a certain domain name.

8.2 The second case: filtering

The second way to protect the DNS server is filtering the requests to the DNS server from the infected hosts to resolve a specific domain. In this case, the DNS server will receive requests, but not fully. Some packets will be blocked by Snort.

Snort should have special rules in the rule file to be able to block packages. This rule should block the requests for specific domain names to the DNS server on port 53, the dedicated port for Domain Name System. And, Snort should not block all packets, but only if the number of packets of the same type exceeds the limit, for example, packets per second. The next rule is consistent with all the necessary requirements:

*drop udp any any -> any 53 (msg:"DNS FLOOD"; *
*content:"severen"; nocase; detection_filter:track by_src, *
count 5, seconds 5; gid:1; sid:10000002; rev:1;)

This rule blocks packets coming from any IP address and from any port to the port 53, and also writes the data into the log file. Snort checks the packages for the presence of the expression *severen*. The rule takes into account only this portion of the domain name, because the rest of the query is generated by the bot. It is a non-existent name and presents as a random set of letters and numbers.

The blocking of packets occurs, if the limit is exceeded - 5 matches in 5 seconds from the source address. It is regulated by the option *detection_filter: track by_src, count 5, seconds 5;*. These values can be changed depending on the frequency of the hosts' suspicious activity and on the security administrator's decision.

As in the previous case, it is enough to have only one rule for the UDP packets. But in the real network, a rule for TCP protocol must be added as follows:

*drop tcp any any -> any 53 (msg:"DNS FLOOD"; *
*content:"severen"; nocase; detection_filter:track by_src, *
count 5, seconds 5; gid:1; sid:10000003; rev:1;)

As it was mentioned above, in the laboratory environment there are only three hosts, and two of them are on the same network. That is why, there is no need to write specific IP addresses. In real terms instead of the *any*, the host address (source address) and the DNS server address (destination address) must be added.

Now, it is time to test the new rule:

1. Tcpdump utility is running with the command:
   tcpdump -n -i enp2s2
2. Next, Snort is configured with the command:
   snort -c /etc/snort/snort.conf -i enp2s1:enp2s2 -Q
3. And finally, the packet generator is running with the command:
   ./botlike

At the end of experiment the next results come up. On the console of the DNS virtual machine there is data about the captured packets and their content (FIGURE 11). Each entry has data about IP addresses and ports (source and destination) and the A request for resolving domain names. For example, the first packet asks for resolving *ilcpskl.severen.net.* And, FIGURE 12 shows the output of the Snort system. Each entry gives the information about actions performed. As tcpdump output, Snort also gives the data about the protocol, source and destination IP addresses and ports used.

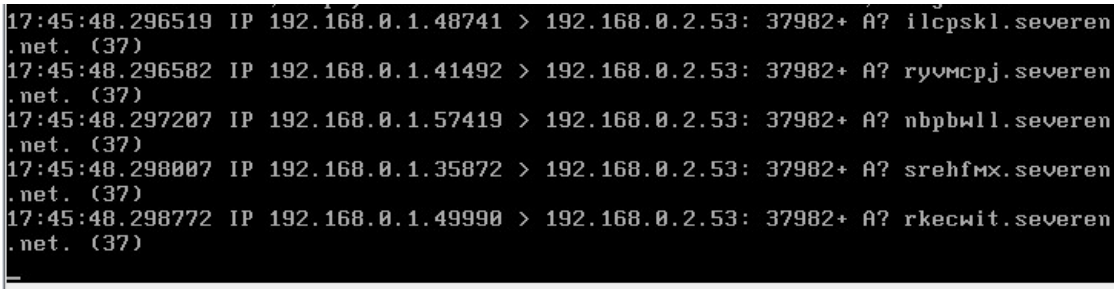


FIGURE 11. Tcpdump output, the second scenario

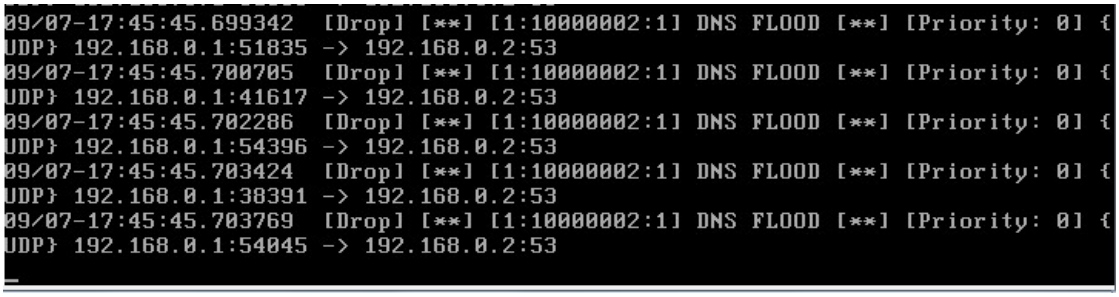


FIGURE 12. Snort IDS output, the second scenario

It was mentioned in the configurations of bot script that the variable CNT has a value of 10. This is the number of packets generated by the program. The value of CNT variable may be changed as needed. And the Snort system starts to drop the packets when there are more than five packets per five seconds. The outputs of tcpdump and Snort

clearly illustrate rule used: five packets reach the destination host and other five packets are blocked by Snort because of the limit.

The difference between the output times (in both scenarios) is a mistake of the devices, as FIGURES 9 to 12 are parts of result screenshots. This can be seen in Appendix C. Time is synchronized at each switching-on, but in a special way.

8.3 Results

These experiments make it possible to imagine the behaviour of the protection system of the DNS server in a real network. Since the main task was to find the most appropriate way to protect the company's DNS service with the use of Snort IDS, it is possible to say that a partial blockage of DNS queries is a more preferable option.

This option is better, since a part of packages still passes and reaches the destination point, and it is likely that these packages are not generated by bot. And since only infected computers are considered, and other users and clients will not be affected by this protection, they will have full access to this domain. The rules must be written only for a narrow range of machines that are listed for suspicious activity.

Also, the rules start to work if the limit is exceeded, and the number of packages sent by the user per time unit is considerably less than the number of packets generated by a bot for the same time. This is confirmed by the experiments. It was written in the rule “five packages for five seconds”, but to achieve this limit, it took only a couple of seconds. At the same time a real user is not able to send so many requests for the same domain in such a short period of time.

In my opinion, the problem is not completely solved. It was necessary to protect DNS servers from the bot-generated requests, and this decision also involved the occasional blocking of users’ requests. Unfortunately, I was not able to configure Snort IDS to carefully sorted packages for “right” and “generated”. Maybe it can be done by using some other tool in addition to the Snort system. Maybe in the future such filtering would be possible, but now single Snort IDS cannot deal with it.

9 CONCLUSION

With this work I was able to further explore the principles and structure of the Domain Name System, learn the architecture of Intrusion Detection System, and had the opportunity to apply the knowledge on practice. The objective of this work was to study the possibilities of the Snort system and its configuration settings to protect the DNS server. The problem is not completely solved at this point. The infrastructure of the Snort system is studied, but the protection of DNS servers is not complete and comprehensive.

One reason is that all the experiments of the practical part were performed only in the laboratory environment. Therefore, it is necessary to further explore the behaviour of the configured protection system in more real conditions before using the results in a real network.

Another reason is the inability of Snort to block only bot-generated packets and to pass true ones. Due to this, it is helpful to try to use various existing Snort plug-ins or additional software that may help to solve the problem fully. Therefore, protection system should be further developed and tested under real conditions before implementation.

One possible way to improve configured system is the implementation of a iptables firewall application in addition to the Snort-inline. In this case, Snort-inline interacts with iptables to receive and to process network traffic. The appropriate iptables rules are used to direct network traffic to the Snort for inspection according to Snort rules. Successful configuration of the whole protection system depends on successful configuration of the iptables.

Despite the limitations which I met during the study, the Snort system is one of the most popular and powerful IDSs on the market today. The system has a huge number of advantages. It is possible to detect (and even prevent) a lot of different suspicious activities. The ability to work with a variety of plug-ins and in several modes, including the inline mode, makes Snort an irreplaceable assistant in the protection system of networks.

BIBLIOGRAPHY

[1] Lam, Kevin 2004. Assessing network security. Washington: Microsoft Press.

[2] Mallery, John 2005. Hardening network security. Osborne: McGraw-Hill.

[3] Panko, Raymond 2004. Corporate computer and network security. New Jersey: Pearson Education.

[4] Pfleeger, Charles 2007. Security in computing 4th edition. New Jersey: Pearson Education.

[5] Starton, Richard 2001. Networking complete 2nd edition. Alameda: Sybex.

[6] Cisco 2015. What Is the Difference: Viruses, Worms, Trojans, and Bots? WWW document. http://www.cisco.com/web/about/security/intelligence/virus-worm-diffs.html. No update information. Referred 10.8.2015.

[7] Zima, Vladimir 2014. Security of global network technologies, 2nd edition. Saint-Petersburg: BHV-Petersburg.

[8] Biryukov, Andrei 2012. Information security: defence and attack. Moscow: DMK-Press.

[9] Hansman, Simon & Hunt, Ray 2004. A taxonomy of network and computer attacks. Computers & Security 3.3.2004, 31-43.

[10] Symantec 2010. What is a Computer Worm? WWW document. http://www.pctools.com/security-news/what-is-a-computer-worm/. No update information. Referred 10.8.2015.

[11] Kaspersky 2015. What is a Trojan Virus? WWW document. http://usa.kaspersky.com/internet-security-center/threats/trojans#.VVxqvvntmko. No update information. Referred 10.8.2015.

[12] Cisco 2015. Snort documents. WWW documents. https://snort.org/documents. No update information. Referred 17.8.2015.

[13] Cisco 2015. Snort User Manual 2.9.7. WWW document. http://manual.snort.org/. No update information. Referred 17.8.2015, 20.8.2015.

[14] Kondratovich, Oleg 2015. Discussions during 13.8.-12.9.2015. Severen-Telecom Ltd.

[15] Ogorkiewicz, Maciej & Frej, Piotr 2002. Analysis of Buffer Overflow Attacks. WWW document. http://www.windowsecurity.com/articles-tutorials/windows_os_security/Analysis_of_Buffer_Overflow_Attacks.html. Updated 8.11.2002. Referred 11.8.2015.

[16] Subramani, Sridhar 2011. Denial of Service attacks and mitigation techniques: Real time implementation with detailed analysis. The SANS Institute. PDF document. https://www.sans.org/reading-room/whitepapers/detection/denial-service-attacks-mitigation-techniques-real-time-implementation-detailed-analysi-33764. No update information. Referred 11.8.2015.

[17] Patrikakis, Charalampos, Masikos, Michalis & Zouraraki, Olga 2004. Distributed Denial of Service Attacks. The Internet Protocol Journal, 13-31.

[18] McDowell, Mindi 2013. Security Tip (ST04-015): Understanding Denial-of-Service Attacks. WWW document. https://www.us-cert.gov/ncas/tips/ST04-015. No update information. Referred 12.8.2015.

[19] Imperva 2015. Denial of Service Attacks. WWW document. https://www.incapsula.com/ddos/ddos-attacks/denial-of-service.html. No update information. Referred 13.8.2015.

[20] Meier, J.D., Mackman, Alex, Dunner, Michael 2003. Improving Web Application Security: Threats and Countermeasures. Microsoft Corporation.

[21] Imperva 2014. Datasheet. Web Attacks: The Biggest Threat to Your Network. PDF document. http://www.imperva.com/docs/ds_web_security_threats.pdf. No update information. Referred 13.8.2015.

[22] Ionescu, Paul 2015. The 10 Most Common Application Attacks in Action. WWW document. http://securityintelligence.com/the-10-most-common-application-attacks-in-action/#.VbvecfntlBc. Updated 8.4.2015. Referred 14.08.2015.

[23] Messmer, Ellen 1998. Facts fight fiction in security. Network World journal, 14.

[24] Rizzo, Tom 2014. 3 Types of Password Security Attacks and How to Avoid Them. WWW document. http://insights.scorpionsoft.com/3-types-of-password-security-attacks-and-how-to-avoid-them. Updated 6.3.2014. Referred 15.8.2014.

[25] Schwarz, Thomas 2004. Buffer Overflow Attack. Santa Clara University. Lecture publications. WWW document.

http://www.cse.scu.edu/~tschwarz/coen152_05/Lectures/BufferOverflow.html. No update information. Referred 16.8.2015.

[26] Matthew, Richard 2001. Intrusion Detection FAQ: Are there limitations of Intrusion Signatures? SANS Institute. WWW document. http://www.sans.org/security-resources/idfaq/limitations.php. Updated 5.4.2001. Referred 16.8.2015.

[27] Masich, Grigorii. DNS - Domain Name System. Institute of Continuous Media Mechanics UB RAS, Perm, Russia. Lecture publications. WWW document. http://www2.icmm.ru/~masich/win/lexion/dns/dns.html. Updated 9.10.2014. Referred 17.8.2015.

[28] The Barking Seal, 2009. Understanding DNS: Essential knowledge for all IT professionals. PDF document. https://www.appliedtrust.com/sites/default/files/assets/resources/understanding-dns-essential.pdf. No update information. Referred 17.8.2015.

[29] Microsoft, 2015. Domain Namespace. TechNet library. WWW document. https://technet.microsoft.com/en-us/library/cc958962.aspx. No update information. Referred 18.8.2015.

[30] Semenov, Yurii 2014. DNS (structure, query processing and resource records). Institute of Theoretical and Experimental Physics. WWW document. http://book.itep.ru/4/44/dns_4412.htm. No update information. Referred 19.8.2015.

[31] Microsoft, 2005. How DNS query works. TechNet library. WWW document. https://technet.microsoft.com/en-us/library/cc775637(v=ws.10).aspx. Updated 21.1.2005. Referred 19.8.2015.

[32] Microsoft, 2005. Understanding zones and zone transfer. TechNet library. WWW document. https://technet.microsoft.com/en-us/library/cc781340(v=ws.10).aspx. Updated 21.1.2005. Referred 19.8.2015.

[33] Kozierok, Charles 2005. DNS Name Resolution Process. The TCP/IP Guide. WWW document. http://www.tcpipguide.com/free/t_DNSNameResolutionProcess-2.htm. Updated 20.9.2005. Referred 20.8.2015.

[34] Vision Quest Integrated Technologies, 2014. How it works: Domain Name to IP Connection & Domain Name Servers. PDF document. http://www.visionquestit.com/html/supportsvs/howitworks/hiw_dns.pdf. No update information. Referred 20.8.2015.

[35] Sun Microsystems, 2000. Solaris Naming Administration Guide. Chapter 28 Introduction to DNS. WWW document. http://docs.oracle.com/cd/E19455-01/806-1387/6jam692f0/index.html. No update information. Referred 21.8.2015.

[36] Semenov, Yurii 2014. DNS Security (DNSSEC). Institute of Theoretical and Experimental Physics. WWW document. http://book.itep.ru/4/4/dnssec.htm. No update information. Referred 22.08.2015.

[37] Beale, Jay, Baker, Andrew R. & Esler, Joel 2007. Snort Intrusion Detection and Prevention Toolkit. Burlington: Syngress Publishing.

[38] Koziol, Jack 2003. Intrusion Detection with Snort. Indianapolis: Sams Publishing.

[39] Caswell, Brian & Beale, Jay 2004. Snort 2.1 Intrusion Detection, 2nd edition. Burlington: Syngress Publishing.

[40] Orebaugh, Angela, Biles, Simon & Babbin, Jacobs 2005. Snort Cookbook. Sebastopol: O'Reilly Media.

[41] Ur Rehman, Rafeeq 2003. Intrusion Detection System with Snort, Apache, MySQL, PHP and ACID. New Jersey: Prentice Hall.

[42] Snort 2014. Configuration files. WWW document. https://github.com/jasonish/snort/tree/2.9.7/etc. No update information. Referred 1.9.2015

[43] Sguil: The Analyst Console for Network Security Monitoring 2014. http://bammv.github.io/sguil/index.html. No update information. Referred 3.9.2015.

[44] NfSen - Netflow Sensor 2011. http://nfsen.sourceforge.net/. No update information. Referred 3.9.2015.

[45] Cacti – the complete RRDTool-based graphic solution 2012. http://www.cacti.net/. No update information. Referred 3.9.2015.

APPENDIX A: SNORT.CONF FILE

```
1   ##################################################
2   # Step #1: Set the network variables.  For more information, see
    README.variables
3   ##################################################
4   ipvar HOME_NET any
5   ipvar EXTERNAL_NET any
6   ipvar DNS_SERVERS $HOME_NET
7   ipvar SMTP_SERVERS $HOME_NET
8   ipvar HTTP_SERVERS $HOME_NET
9   ipvar SQL_SERVERS $HOME_NET
10  ipvar TELNET_SERVERS $HOME_NET
11  ipvar SSH_SERVERS $HOME_NET
12  ipvar FTP_SERVERS $HOME_NET
13  ipvar SIP_SERVERS $HOME_NET
14  portvar HTTP_PORTS
    [36,80,81,82,83,84,85,86,87,88,89,90,311,383,555,591,593,631,801,808
    ,818,901,972,1158,1220,1414,1533,1741,1830,1942,2231,2301,2381,25
    78,2809,2980,3029,3037,3057,3128,3443,3702,4000,4343,4848,5000,51
    17,5250,5600,5814,6080,6173,6988,7000,7001,7005,7071,7144,7145,75
    10,7770,7777,7778,7779,8000,8001,8008,8014,8015,8020,8028,8040,80
    80,8081,8082,8085,8088,8090,8118,8123,8180,8181,8182,8222,8243,82
    80,8300,8333,8344,8400,8443,8500,8509,8787,8800,8888,8899,8983,90
    00,9002,9060,9080,9090,9091,9111,9290,9443,9447,9710,9788,9999,10
    000,11371,12601,13014,15489,19980,29991,33300,34412,34443,34444,
    40007,41080,44449,50000,50002,51423,53331,55252,55555,56712]
15  portvar SHELLCODE_PORTS !80
16  portvar ORACLE_PORTS 1024:
17  portvar SSH_PORTS 22
18  portvar FTP_PORTS [21,2100,3535]
19  portvar SIP_PORTS [5060,5061,5600]
20  portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]
21  portvar GTP_PORTS [2123,2152,3386]
22  ipvar AIM_SERVERS
    [64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0
    /24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.1
    88.153.0/24,205.188.179.0/24,205.188.248.0/24]
23  var RULE_PATH ../rules
24  var SO_RULE_PATH ../so_rules
25  var PREPROC_RULE_PATH ../preproc_rules
26  var WHITE_LIST_PATH /etc/snort/rules
27  var BLACK_LIST_PATH /etc/snort/rules
28  ##################################################
```

```
29  # Step #2: Configure the decoder.  For more information, see README.decode
30  ##################################################
31  config disable_decode_alerts
32  config disable_tcpopt_experimental_alerts
33  config disable_tcpopt_obsolete_alerts
34  config disable_tcpopt_ttcp_alerts
35  config disable_tcpopt_alerts
36  config disable_ipopt_alerts
37  config checksum_mode: all
38  # Configure DAQ related options for inline operation. For more information, see README.daq
39  config daq: afpacket
40  config daq_dir: /usr/local/lib/daq
41  config daq_mode: inline
42  config daq_var: buffer_size_mb=256
43  ##################################################
44  # Step #3: Configure the base detection engine.  For more information, see README.decode
45  ##################################################
46  config pcre_match_limit: 3500
47  config pcre_match_limit_recursion: 1500
48  config detection: search-method ac-split search-optimize max-pattern-len 20
49  config event_queue: max_queue 8 log 5 order_events content_length
50  config paf_max: 16000
51  ##################################################
52  # Step #4: Configure dynamic loaded libraries.
53  # For more information, see Snort Manual, Configuring Snort - Dynamic Modules
54  ##################################################
55  dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
56  dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
57  dynamicdetection directory /usr/local/lib/snort_dynamicrules
58  ##################################################
59  # Step #5: Configure preprocessors
60  # For more information, see the Snort Manual, Configuring Snort - Preprocessors
61  ##################################################
62  # Inline packet normalization. For more information, see README.normalize
63  preprocessor normalize_ip4
```

```
64  preprocessor normalize_tcp: block, rsv, pad, urp, req_urg, req_pay, req_urp, ips, ecn stream
65  preprocessor normalize_icmp4
66  preprocessor normalize_ip6
67  preprocessor normalize_icmp6
68  config policy_mode: inline
69  preprocessor frag3_global: max_frags 65536
70  preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180
71  preprocessor stream5_global: track_tcp yes, \
72    track_udp yes, \
73    track_icmp no, \
74    max_tcp 262144, \
75    max_udp 131072, \
76    max_active_responses 2, \
77    min_response_seconds 5
78  preprocessor stream5_tcp: policy windows, detect_anomalies, require_3whs 180, \
79    overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
80     ports client 21 22 23 25 42 53 70 79 109 110 111 113 119 135 136 137 139 143 \
81        161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070 6665 6666 6667 6668 6669 \
82        7000 8181 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \
83     ports both 36 80 81 82 83 84 85 86 87 88 89 90 110 311 383 443 465 563 555 591 593 631 636 801 808 818 901 972 989 992 993 994 995 1158 1220 1414 1533 1741 1830 1942 2231 2301 2381 2578 2809 2980 3029 3037 3057 3128 3443 3702 4000 4343 4848 5000 5117 5250 5600 5814 6080 6173 6988 7907 7000 7001 7005 7071 7144 7145 7510 7802 7770 7777 7778 7779 \
84        7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914 7915 7916 \
85        7917 7918 7919 7920 8000 8001 8008 8014 8015 8020 8028 8040 8080 8081 8082 8085 8088 8090 8118 8123 8180 8181 8182 8222 8243 8280 8300 8333 8344 8400 8443 8500 8509 8787 8800 8888 8899 8983 9000 9002 9060 9080 9090 9091 9111 9290 9443 9447 9710 9788 9999 10000 11371 12601 13014 15489 19980 29991 33300 34412 34443 34444 40007 41080 44449 50000 50002 51423 53331 55252 55555 56712
86  preprocessor stream5_udp: timeout 180
87  preprocessor http_inspect: global iis_unicode_map unicode.map 1252 compress_depth 65535 decompress_depth 65535
88  preprocessor http_inspect_server: server default \
```

```
89      http_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS BITS_POST CCM_POST SMS_POST RPC_IN_DATA RPC_OUT_DATA RPC_ECHO_DATA } \
90      chunk_length 500000 \
91      server_flow_depth 0 \
92      client_flow_depth 0 \
93      post_depth 65495 \
94      oversize_dir_length 500 \
95      max_header_length 750 \
96      max_headers 100 \
97      max_spaces 200 \
98      small_chunk_length { 10 5 } \
99      ports { 80 81 311 383 591 593 901 1220 1441 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000 7001 7144 7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8123 8180 8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 11371 34443 34444 41080 50002 55555 } \
100     non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
101     enable_cookie \
102     extended_response_inspection \
103     inspect_gzip \
104     normalize_utf \
105     unlimited_decompress \
106     normalize_javascript \
107     apache_whitespace no \
108     ascii no \
109     bare_byte no \
110     directory no \
111     double_decode no \
112     iis_backslash no \
113     iis_delimiter no \
114     iis_unicode no \
115     multi_slash no \
116     utf_8 no \
117     u_encode yes \
118     webroot no
119   preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no_alert_multiple_requests no_alert_large_fragments no_alert_incomplete
120   preprocessor bo
```

```
121 preprocessor ftp_telnet: global inspection_type stateful encrypted_traffic
    no check_encrypted
122 preprocessor ftp_telnet_protocol: telnet \
123   ayt_attack_thresh 20 \
124   normalize ports { 23 } \
125   detect_anomalies
126 preprocessor ftp_telnet_protocol: ftp server default \
127   def_max_param_len 100 \
128   ports { 21 2100 3535 } \
129   telnet_cmds yes \
130   ignore_telnet_erase_cmds yes \
131   ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP }
    \
132   ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
133   ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
134   ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST }
    \
135   ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
136   ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
137   ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
138   ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD }
    \
139   ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ
    XSEM } \
140   ftp_cmds { XSEN XSHA1 XSHA256 } \
141   alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV
    NOOP PASV PWD QUIT REIN STOU SYST XCUP XPWD } \
142   alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR
    RNFR STOR STOU XMKD } \
143   alt_max_param_len 256 { CWD RNTO } \
144   alt_max_param_len 400 { PORT } \
145   alt_max_param_len 512 { SIZE } \
146   chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD
    } \
147   chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
148   chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD
    } \
149   chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT
    } \
150   chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE }
    \
151   chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER }
    \
```

```
152     chk_str_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD
        XRSQ } \
153     chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \
154     cmd_validity ALLO < int [ char R int ] > \
155     cmd_validity EPSV < [ { char 12 | char A char L char L } ] > \
156     cmd_validity MACB < string > \
157     cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
158     cmd_validity MODE < char ASBCZ > \
159     cmd_validity PORT < host_port > \
160     cmd_validity PROT < char CSEP > \
161     cmd_validity STRU < char FRPO [ string ] > \
162     cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [ num-
        ber ] } >
163   preprocessor ftp_telnet_protocol: ftp client default \
164     max_resp_len 256 \
165     bounce yes \
166     ignore_telnet_erase_cmds yes \
167     telnet_cmds yes
168   preprocessor smtp: ports { 25 465 587 691 } \
169     inspection_type stateful \
170     b64_decode_depth 0 \
171     qp_decode_depth 0 \
172     bitenc_decode_depth 0 \
173     uu_decode_depth 0 \
174     log_mailfrom \
175     log_rcptto \
176     log_filename \
177     log_email_hdrs \
178     normalize cmds \
179     normalize_cmds { ATRN AUTH BDAT CHUNKING DATA
        DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY } \
180     normalize_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX
        QUEU QUIT RCPT RSET SAML SEND SOML } \
181     normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB
        VRFY X-ADAT X-DRCP X-ERCP X-EXCH50 } \
182     normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR
        XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
183     max_command_line_len 512 \
184     max_header_line_len 1000 \
185     max_response_line_len 512 \
186     alt_max_command_line_len 260 { MAIL } \
187     alt_max_command_line_len 300 { RCPT } \
188     alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
```

```
189     alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT
        DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \
190     alt_max_command_line_len 246 { SEND SAML SOML AUTH
        TURN ETRN DATA RSET QUIT ONEX QUEU STARTTLS TICK
        TIME TURNME VERB X-EXPS X-LINK2STATE XADR XAUTH
        XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
191     valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG
        EHLO EMAL ESAM ESND ESOM ETRN EVFY } \
192     valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX
        QUEU QUIT RCPT RSET SAML SEND SOML } \
193     valid_cmds { STARTTLS TICK TIME TURN TURNME VERB
        VRFY X-ADAT X-DRCP X-ERCP X-EXCH50 } \
194     valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR
        XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
195     xlink2state { enabled }
196   preprocessor ssh: server_ports { 22 } \
197             autodetect \
198             max_client_bytes 19600 \
199             max_encrypted_packets 20 \
200             max_server_version_len 100 \
201             enable_respoverflow enable_ssh1crc32 \
202             enable_srvoverflow enable_protomismatch
203   preprocessor dcerpc2: memcap 102400, events [co ]
204   preprocessor dcerpc2_server: default, policy WinXP, \
205     detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \
206     autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \
207     smb_max_chain 3, smb_invalid_shares ["C$", "D$", "ADMIN$"]
208   preprocessor dns: ports { 53 } enable_rdata_overflow
209   preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 5061
        7801 7802 7900 7901 7902 7903 7904 7905 7906 7907 7908 7909 7910
        7911 7912 7913 7914 7915 7916 7917 7918 7919 7920 }, trustservers,
        noinspect_encrypted
210   preprocessor sensitive_data: alert_threshold 25
211   preprocessor sip: max_sessions 40000, \
212     ports { 5060 5061 5600 }, \
213     methods { invite \
214           cancel \
215           ack \
216           bye \
217           register \
218           options \
219           refer \
220           subscribe \
221           update \
```

```
222                 join \
223                 info \
224                 message \
225                 notify \
226                 benotify \
227                 do \
228                 qauth \
229                 sprack \
230                 publish \
231                 service \
232                 unsubscribe \
233                 prack }, \
234           max_uri_len 512, \
235           max_call_id_len 80, \
236           max_requestName_len 20, \
237           max_from_len 256, \
238           max_to_len 256, \
239           max_via_len 1024, \
240           max_contact_len 512, \
241           max_content_len 2048
242         preprocessor imap: \
243           ports { 143 } \
244           b64_decode_depth 0 \
245           qp_decode_depth 0 \
246           bitenc_decode_depth 0 \
247           uu_decode_depth 0
248         preprocessor pop: \
249           ports { 110 } \
250           b64_decode_depth 0 \
251           qp_decode_depth 0 \
252           bitenc_decode_depth 0 \
253           uu_decode_depth 0
254         preprocessor modbus: ports { 502 }
255         preprocessor dnp3: ports { 20000 } \
256           memcap 262144 \
257           check_crc
258         preprocessor reputation: \
259           memcap 500, \
260           priority whitelist, \
261           nested_ip inner, \
262           whitelist $WHITE_LIST_PATH/white_list.rules, \
263           blacklist $BLACK_LIST_PATH/black_list.rules
264         ##################################################
265         # Step #6: Configure output plugins
```

```
# For more information, see Snort Manual, Configuring Snort - Output Modules
##################################################
output alert_fast: stdout
include classification.config
include reference.config
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
# NOTE: All categories are enabled in this conf file
##################################################
include /etc/snort/rules/local.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/dos.rules
include threshold.conf
```

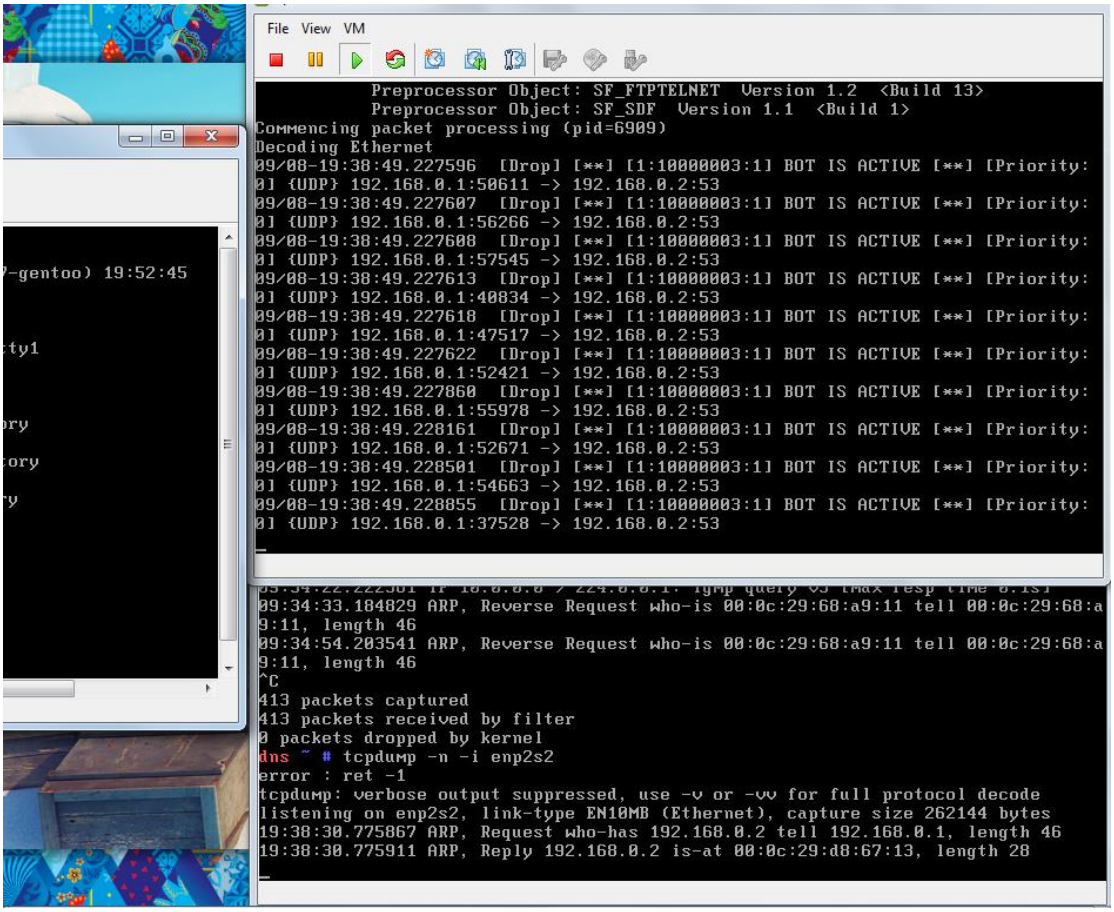
APPENDIX B: EXECUTABLE SCRIPT (ATTACKER MACHINE)

```
1	#include <stdio.h>
2	#include <string.h>
3	#include <stdlib.h>
4	#include <arpa/inet.h>
5	#include <unistd.h>
6	#include <sys/socket.h>
7
8	#define SLP 10000
9	#define MAX 128
10	#define PORT 53
11	#define HOST “192.168.0.2”
12	#define CNT 10
13	#define REC “severen.net”
14
15	int main(void) {
16	  int s, i, j, k;
17	  struct sockaddr in so;
18	  const char h[]="945E010000010000000000007", t[]="0000010001";
19	  // last '07' is next label lenght so i use 7-th chars variable label
20
21	  char b[MAX], c[3], q[MAX]="\0";
22
23	  for(j=0; j,CNT; j++) {
24	    s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
25	    memset((char*)&so, 0, sizeof(so));
26	    so,sin_family=AF_INET;
27	    so.sin_port=htons(PORT); inet_aton(HOST,&so.sin_addr);
28	    strcpy(q,h); //start with header (ip)+(udp)
29	    //variable label
30	    for(i=0; i<7; i++) {
31	        sprintf(c, "%02x", (97+rand()%25));
32	        //printf("%s\n", c)
33	        strcat(q, c);
34	        }
35
36	    sprintf(c, '%02x", 7); //inserts
37	    strcat(q, c); //delimiter for next 7th 'severen'
38	    memcpy(&b[0], REC, strlen(REC)); //static label
39	    b[strlen(REC)]=0; //terminate b[]
40	    for(i=0; i<strlen(REC); i++) {
41	        //printf(char=%c hex=%02x\n", b[i], b[i]);
42	        if(b[i]=='.') strcpy(c, "03");
```

```
43                      else sprintf(c, "%02x", (int)b[i]);
44                      strcat(q, c);
45                      } //concatenate hex RR
46              strcat(q, t); //add trailer (end of RR)
47              //printf("%s\n", q);
48              //exit(1);
49              for(i=0, k=0; i<strlen(q); i+=2, k++) {
50                      memcpy(&c, &q[i], 2); c[2]=0;
51                      b[k]=(int)strtol(c, NULL, 16);
52                      //printf("%c%c\n", q[i], q[i+1]);
53                      }
54              sendto(s, b, k, 0, (struct sockaddr*)&so, sizeof(so));
55              close(s);
56              usleep(SLP);
57              }
58          return 0;
59      }
```

APPENDIX C: RESULT SCREENSHOTS

FIRST SCENARIO OUTPUT



SECOND SCENARIO OUTPUT