Fake Packet Generation, Detection and its analysis using Network Security

Nikunj Pansari
Computer Science and Engineering
Indian Institute of Technology, Dharwad

Rishit Saiya
Computer Science and Engineering
Indian Institute of Technology, Dharwad

***Abstract*—The threat of network Trojans looms largely on mission-critical applications. This research work illustrates the generation of fake packets distinct from general conventional network traffic and their detection using tools like Scapy, Snort, and simulation of an IDS (Intrusion Detection System). It caters to the execution of the Payload (packet generation) and its real-time analytical understanding based on MITM (Man-in-the-middle) attack and its illustration using Scapy, Wireshark & Snort, thus utilizing the network analysis techniques. Furthermore, Port Security strategies to mitigate the most-vulnerable threats are also defined, possible DNA cryptographic techniques, and state-of-the-art Quantum Cryptography is also explored comprehensively.**

***Index Terms* - Network Security, Packet Analysis, Fake Packet Generation, Port Security, DNA Cryptography, Quantum Cryptography.**

I. Introduction

In recent times, most of the devices and appliances are now connected under a network where they can communicate with each other using a set of rules and networking compliance. This opens up the surface area of attack on such network that instigates a fatal MITM(Man-in-the-middle) attack [4][6].The Computer Network proves to be a substantial and integral component for packet analysis, following which, the communication protocols are defined over different interacting entities for the goal of resource sharing and information transfer. Now, this communication is illustrated through different pathways of communication termed as 'Network Topologies', which can be utilized for several real-time telecommunication network concepts, either based on optical, wired, or radio frequency with wireless technologies [1].

This research work aims to illustrate relevant demonstration on how network traffic (in form of packets) can be obfuscated using fake packet generation, which is further elaborated on the detection of such fake packets in the network as a mitigation technique, thus quite substantial in long run for implementation and ensuring strongest security (adopting most secure cryptographic algorithms for encryption & decryption).

Subsequently, illustrated the implementation and analysis using tools like Snort, Scapy, Wireshark, and Nmap, which were quite beneficial and at the same time, quite versatile in inferring the obtained results [23]. Two different kinds of approaches were also proposed for scanning the packets before and after detection using Nmap and snort, and the other one using snort and Wireshark for inferring the sniffing results for the obtained packets. Optimized and advanced implementation of the fake packet generation and detection was carried pertaining to the real-time specifications and circumstances from attacker's and target's side point of view, in which Configurations were defined for utilized tools i.e. snort and Scapy for packet detection, following which illustrated the encapsulation of packets using Snort & Scapy. Defined whitelist and blacklist traffic for effective comparative analysis of packet detection efficiency across Scapy & Snort [22].

Reconnaissance on the Blacklisted IPs was demonstrated which included illustrations on Blacklisted IPs, thus resulting in inferring the results as graphs exhibiting the Blacklist IP traffic functioning using two different scenarios.

Not only constraint to that, optimized techniques in terms of security, less computation time, and more computational efficiency ensured through certain well-known techniques such as Port Security, which also included various violation modes of port security, that is further helpful in inferring possible feasible approach for mitigating the risk of occurrence of attacks using open ports using Port-knocking (implemented through Secure Port Knock-Tunneling (SPKT)). Also, a Port Facility Security Plan (PFSP) was adopted as a strategy in EU nations for mitigating port-related threats [9][10].

The state-of-the-art techniques of DNA Cryptography, which involves computation using different nitrogenous bases, to improve efficacy and parallelism, maybe at the cost of complexity, sometimes mostly involving DNA hybridization and a one-time padding scheme [13].

Further, to perform illustrations on cryptography involving quantum mechanics namely Quantum Cryptography was reviewed involving the computations of pulse rather than individual photons, which mainly involved the implementation of Quantum Key Distribution (QKD). Then, various attack possible strategies were discussed to get a viewpoint on the source and type of attacks that may be useful in mitigating the risk in critical systems. Additionally, some optimized and highly secured existing works were helpful in inferring the establishment of a highly authenticated and optimized system implementation approach that could significantly be very helpful in defining less-storage and more efficient network systems, involving fewer probabilities of vulnerabilities detection [17][19].

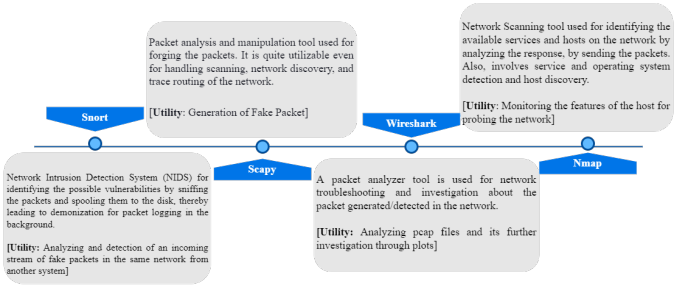


Fig. 1. Analysis of Tools [1][3][7][20]

II. Analysis of Components

A. Fake Packet Injection

Fake Packets are defined as if, they appear as a source originating out of the normal communicating system. Now, fake packet injection is defined through the establishment of random packets which are assumed to be defined as a part of the generally used communication system. This concept leverages the security of the system as it permits or provides the path for the intruder or attacker to intercept the packets (either through spoofing or forging), thus compromising and degrading the security of the system [6][2].

B. Real-time applications of Fake Packet

Notable real-time applications utilizing the fake packet concept include DOS (Denial of Service) attack and MITM (Man-in-the-Middle) attack . In a MITM attack, the malicious or suspicious piece of code is analyzed by the attacker, the intruder for leveraging the security of the system, and by a reverse software engineer for identifying the possible threats or vulnerabilities in their network systems [Fig 1]. While executing the DOS attack the attacker uses the fake packets as means to disrupt the entire services or functioning of the network system, thereby may lead to degradation of the entire system infrastructure [4][8].

C. Advanced Detection Mechanisms from Raw Packet Capture

The characteristics of network traffic define the pathway for efficient network traffic analysis. Some notable features substitute the OSI model namely Protocols used, Source & Destination IP, Size of Packet, Source & Destination Ports, Payloads & flags. All these features cater to useful information about the Network IDS [8].

Nowadays, many corporate switches are useful in exporting NetFlow, raw data, sFlow, or related kinds of features. Mainly emphasized on NetFlow, which defines the Source and Destination port, and transferred amount of traffic per flow. Considering the pragmatic approach for real-time use cases, it is useful for analysing and monitoring the payload for packets transmitted or generated in the network, but it requires more complex computations [3].

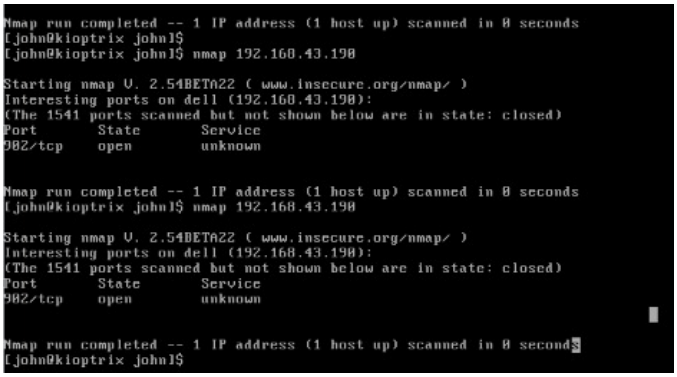


Fig. 2. Nmap Scan (Kioptrix OS) before detection

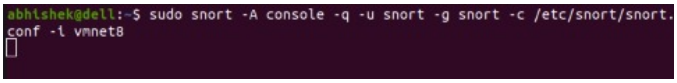


Fig. 3. Snort Scan (Ubuntu OS) before detection

III. Approaches

A. Initial Approach

To get a good overview regarding packet analysis and packet generation, the implementation of simple IDS is illustrated below.

Implementation: For better implementation, utilize Kioptrix as the required Virtual Machine (VM) and can choose any of the Ubuntu versions (OS) for analysis, which consolidated constitute the system setup, where the VM was set as a device connected to the same LAN (Local Area Network) of the Ubuntu OS. Integration of Nmap and Snort as IDS in the above-mentioned OS respectively was carried out. Mainly emphasized to test the shortcomings of packet detection for various IDS connected to the internet for devices in the same network as well as remote network. [5][20].

So, essentially the packet is sniffed in the other active system on the network using the IDS integrated with the detection system. The illustration shows that the IDS were switched to sniffing mode [Fig 2 and 3].

Fig 4 and 5, define that each active system was detected as traffic in their respective IDS.

Inference: With this small rather effective approach, the outcomes demonstrate the fake packet and detection through a remote system to the desired network that is constrained with existing conditions and modern state-of-the-art security measures on a local network only. So, to further escalate the implementation, the final path for obfuscation implementation was built.

B. Final Approach

With some concepts grasped from the Initial Approach and many trials within the setup and selection of tools, finally

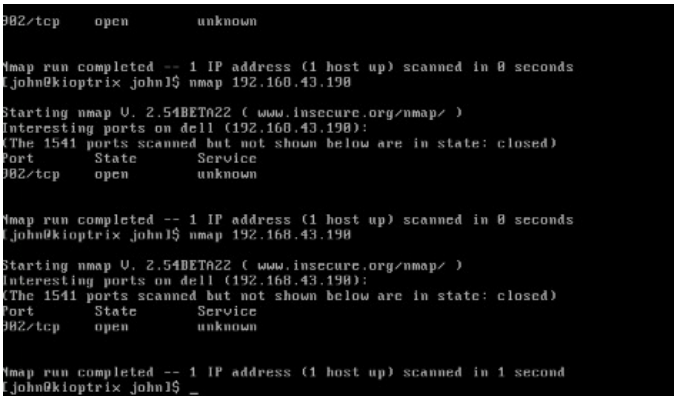


Fig. 4. Nmap Scan (Kioptrix OS) after detection

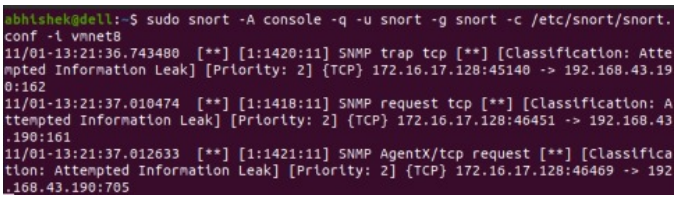


Fig. 5. Snort Scan (Ubuntu OS) after detection

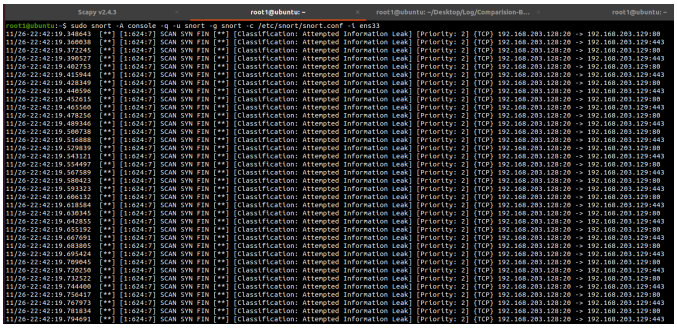


Fig. 6. Snort Sniffing Results - Initial Test

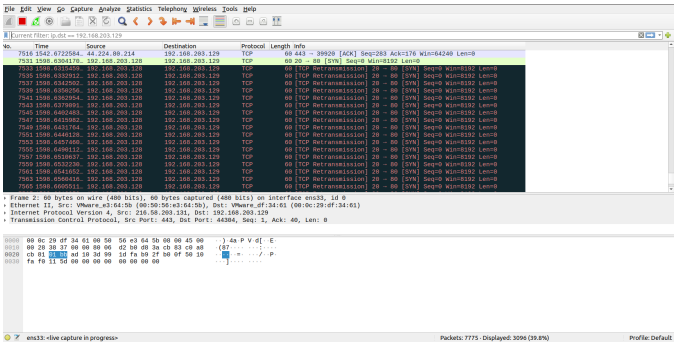


Fig. 7. Wireshark Sniffing Results - Initial Test

inferred to ponder upon the following implementation.

Implementation: Initiated by deploying 2 unique operating systems (Ubuntu) in a Virtual Machine environment. This was done to ensure that all the devices involved in implementation are connected to same LAN.

C. Nomenclature & Environment Setup

Integrated Scapy in OS1 (here on referred to as Attacker) and Snort in OS2 (here on referred as Target) as the environment setting up step.

D. Fake Packet Generation (Phase-I)

As illustrated before, the analysis was carried out using Scapy for generating packets and thus helps in customizing the protocols to experiment with the scope of generation as well as the IDS at the Target. WLOG, the sequence of generation of the packet was done at the Attacker's end. So, initially began with some standard packet generation from Attacker and sending to Target. The stated command used to generate a simple packets stream:

```
send(IP(dst="<dst IP>")/TCP(),count=50)
```

E. Fake Packet Detection (Phase-I)

At the target's side, utilized Snort to detect an incoming stream of fake packets which were sent by the Attacker. [Fig 6] Illustration shows the Snort detection of stream of fake packets generated from the attacker's end. To be affirmative about the authenticity of Snort results, sniffing of the traffic using Wireshark was performed. The illustration shows that Wireshark sniffed traffic of fake packets generated from the attacker's end [Fig 7] . The following command was used to detect a simple packet stream on the Target end:

```
sudo snort -A console -q -u snort -g snort
-c /etc/snort/snort.conf -i ens33
```

IV. Advanced Implementation

For this implementation, further monitoring and analysis had to mimic the whole generation of the fake packet and its detection in attackers and target sides respectively.

Now, that the illustration showed that the fake packet can be generated in a computer network and can be detected as well, for optimization and testing, some advanced and novel techniques to detect/blacklist the fake packets in a network were proposed [1][3].

A. Configuration of Snort & Scapy for Detection

Snort: As Snort is an open-source utility, so altered the rules in configuration files named `snort.conf`, pertaining to the use case and requirements [5][24][25].

Snort was allowed authorized access to the file containing chunks of IP address to get an insight about the behavior and analysis about trusted hosts and vulnerable hosts. Now, the whitelists are meant to store the trusted IP addresses and the blacklists pertain to the storage of malicious IP addresses. In the standard installations of Snort, the configuration file is placed at **`/etc/snort/snort.conf`**. In the `snort.conf`, initiated configuring `preprocessor reputation` as follows:

```
preprocessor reputation: \
  memcap 500, \
  priority whitelist, \
  nested_ip inner, \
  scan_local, \
  whitelist $WHITE_LIST_PATH/white_list.rules
blacklist $BLACK_LIST_PATH/black_list.rules
```

Also, configured WHITE_LIST_PATH & BLACK_LIST_PATH as well. Implemented as follows (Consider `/etc/snort/rules/iplists` as X):

```
var WHITE_LIST_PATH X
var BLACK_LIST_PATH X
```

Now that the subnets declared, were stored in `white_list.rules` & `black_list.rules` files, so, defined the absolute path for the files as follows:

```
sudo mkdir X
sudo touch X/black_list.rules
sudo touch X/white_list.rules
```

Scapy: Scapy does not utilise an external configuration for the requirements, hence doesn't require any modifications [22].

V. PCAP Encapsulation in Snort & Scapy

The main task emphasized comparing and analyzing various detection techniques/algorithms which Scapy & Snort were utilizing. Additionally, defined the comparison between the capacity of packets detection over different time intervals for Snort vs Scapy, to test its versatility.

For initiating that step, need to be assured that a proper way to generate `pcap` files in Snort as well as Scapy, such that can be helpful in analysing later `pcap` files in Wireshark [3][7].

A. Snort PCAP Encapsulation

Snort stores all its log files at **`/var/log/snort`** with filename as `snort.log.<timestamp>` with `pcap` filetype. Thus, extracted the file using timestamp and further analyzed it in Wireshark.

B. Scapy PCAP Encapsulation

After sniffing all the packets, executing the following commands to save the output as `pcap` file to further analyze in Wireshark:

```
a = _
wrpcap("test.pcap", a)
```

VI. Comparison in Detection Efficiency across Scapy & Snort

A. *Whitelist Traffic*

This is a type of traffic which is not blacklisted by Target system and Target's end allows all types of traffic interceptions to that particular subnet in the computer network.

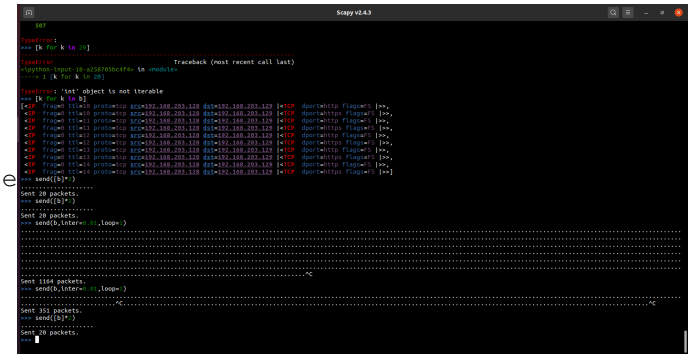


Fig. 8. Scapy - Sending fake TCP packets

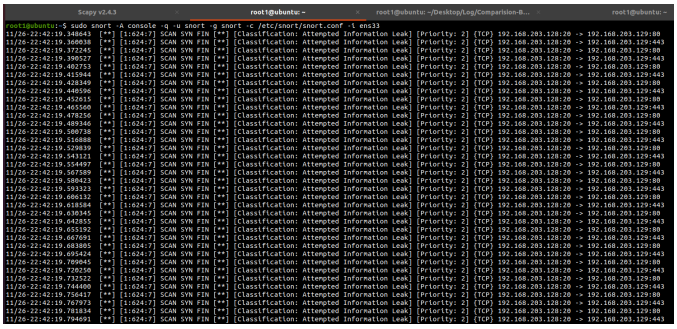


Fig. 9. Snort Sniffing Results - Whitelist Traffic

*1) **Snort Detection**:* Executed following command for interception of traffic: (Consider `/etc/snort/snort.conf` as X)

```
sudo snort -A console -q -c X -i ens33
```

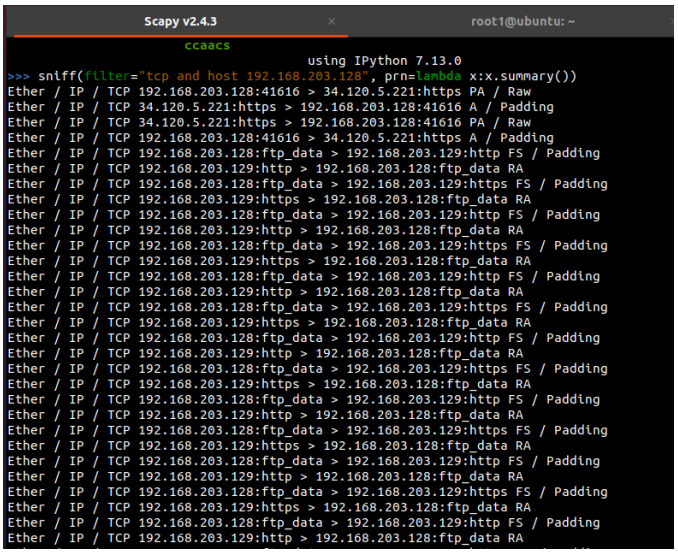
*2) **Scapy Detection**:* For proper traffic interception, commands executed are defined below:

```
sniff(filter="tcp and host <src IP>",
prn=lambda x:x.summary())
```

Now, the fake packet detection (Phase-I) caters to similar behavior to whitelist traffic, thus Scapy and Snort packet detection analysis would yield similar outcomes, which is in contrast to the outcomes from whitelist traffic which are trivial. Then, interception of the whitelist traffic was observed to the target's end from the attacker's site. Encapsulation of the findings in `pcap` file were executed by strategy defined as in *PCAP Encapsulation in Snort & Scapy* section.

Furthermore, Fig 11 & 12 illustrates the graphical analysis of `pcap` files in Wireshark for the Number of Packets received in intervals of time for Snort & Sniff IDS for Whitelist traffic.

*3) **Comparison - Whitelist Traffic**:* Both Scapy and Snort inferred good results in detecting the network traffic. Now, there were several protocols integrated into Snort, due to which it is slightly less efficient in terms of packet analysis as compared to Scapy in detection [22][23]. But because of the configurations defined for these protocols, Snort doesn't



```
                          ccaacs
                                            using IPython 7.13.0
>>> sniff(filter="tcp and host 192.168.203.128", prn=lambda x:x.summary())
Ether / IP / TCP 192.168.203.128:41616 > 34.120.5.221:https PA / Raw
Ether / IP / TCP 34.120.5.221:https > 192.168.203.128:41616 A / Padding
Ether / IP / TCP 34.120.5.221:https > 192.168.203.128:41616 PA / Raw
Ether / IP / TCP 192.168.203.128:41616 > 34.120.5.221:https A / Padding
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
```

Fig. 10. Scapy Sniffing Results - Whitelist Traffic

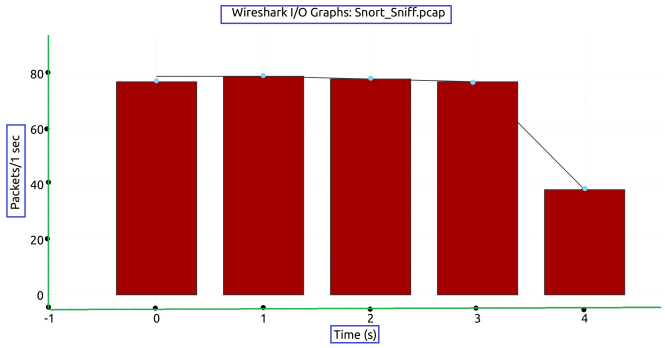


Fig. 11. PCAP Analysis - Snort IDS - Whitelist Traffic

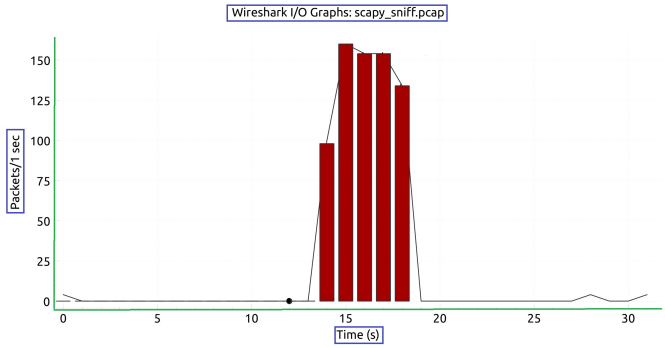
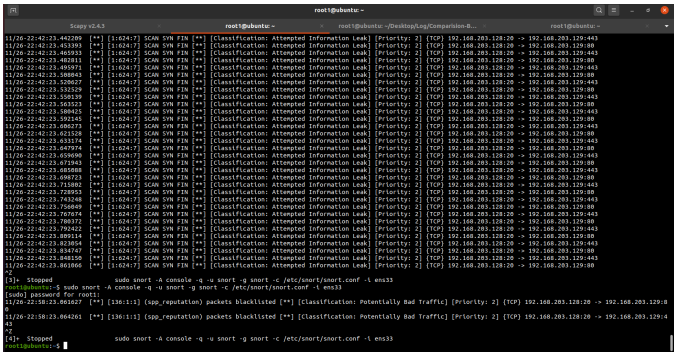


Fig. 12. PCAP Analysis - Scapy IDS - Whitelist Traffic



```
11/26-22:42:23.442209  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.453393  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.465933  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.482811  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.495971  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.508043  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.520627  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.532529  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.550139  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.563523  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.580425  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.592145  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.606273  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.621528  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.633174  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.647974  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.659690  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.671943  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.685088  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.698723  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.715802  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.728953  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.743248  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.756049  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.767674  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.780372  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.792422  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.809114  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.823054  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.834747  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
11/26-22:42:23.848150  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:443
11/26-22:42:23.861066  [**] [1:624:7] SCAN SYN FIN [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:80
^Z
[3]+  Stopped                 sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
root1@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
[sudo] password for root1:
11/26-22:58:23.061627  [**] [136:1:1] (spp_reputation) packets blacklisted [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:8
0
11/26-22:58:23.064261  [**] [136:1:1] (spp_reputation) packets blacklisted [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.203.128:20 -> 192.168.203.129:4
43
^Z
[4]+  Stopped                 sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33
root1@ubuntu:~$ 
```

Fig. 13. Snort Sniffing Results - Blacklist Traffic

show every packet that is being sent or received. It shows the packets on the priority level of intrusion [24][25].

Scapy doesn't provide leverage to easily analyze the packet behavior consistently. As illustrated [Fig 8], it can be inferred that there were 351 packets being sent through Scapy, and subsequently, the plot states that Scapy has detected way more than a number of packets sent [Fig 12]. Those other extra packets detected in Scapy are from other traffic under the same network. On the contrary, Snort only shows 351 packets because of the configuration of rules done in Snort [Fig 11].

B. *Blacklist Traffic*

This is a type of traffic that is blacklisted on a Target system and Target's end doesn't allow all types of traffic interceptions to that particular subnet in the computer network. This is a mitigation technique to non-trusted sources which are under the same network and can cause a potential threat to subnets, IoT Devices, other systems under the network.

*1) **Snort Detection**:* Executing following command for interception of traffic. (Consider `/etc/snort/snort.conf` as X)

```
sudo snort -A console -q -c X -i ens33
```

Since the Blacklist traffic is different, the Snort Detection also blacklists those packets [Fig 13]. It also depicts the potential threat of incoming traffic packets on Snort IDS.

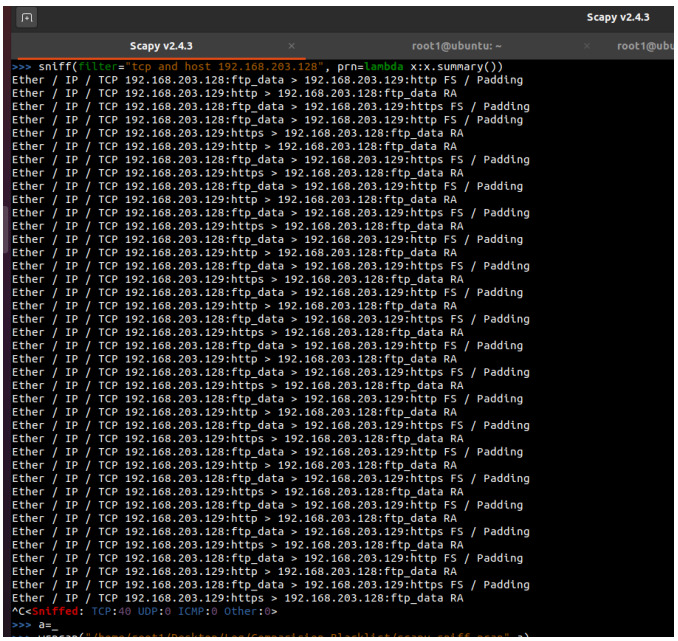
*2) **Scapy Detection**:* Executing following command for interception of traffic:

```
sniff(filter="tcp and host <src IP>",
prn=lambda x:x.summary())
```

Furthermore, Fig 15 & 16 illustrates the graphical analysis of `pcap` files in Wireshark for Number of Packets received in intervals of time for Snort & Sniff IDS for Blacklist traffic.

*3) **Comparison - Blacklist Traffic**:* Unlike Whitelist Traffic here, because of the inclusion of the blacklist traffic rule in Snort, it detects and blocks the incoming packets after identifying the packets to be from blacklisted IP addresses. As a consequence, only 2 packets were received out of 20 packets sent [Fig 14 & 9].

On the other hand, Scapy does not have such scope of freedom and hence it detects all the packets being sent from



```
>>> sniff(filter="tcp and host 192.168.203.128", prn=lambda x:x.summary())
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:http FS / Padding
Ether / IP / TCP 192.168.203.129:http > 192.168.203.128:ftp_data RA
Ether / IP / TCP 192.168.203.128:ftp_data > 192.168.203.129:https FS / Padding
Ether / IP / TCP 192.168.203.129:https > 192.168.203.128:ftp_data RA
^C<Sniffed: TCP:40 UDP:0 ICMP:0 Other:0>
>>> a=_
```

Fig. 14. Scapy Sniffing Results - Blacklist Traffic

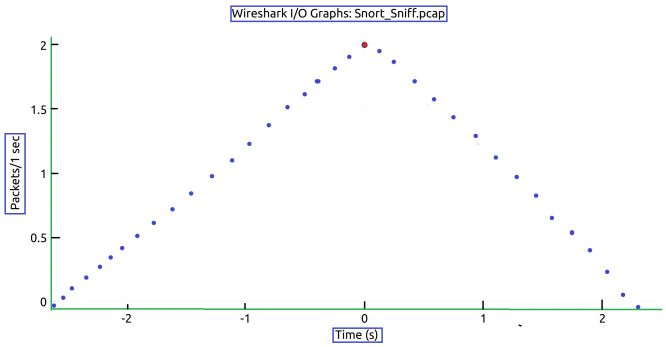


Fig. 15. PCAP Analysis - Snort IDS - Blacklist Traffic

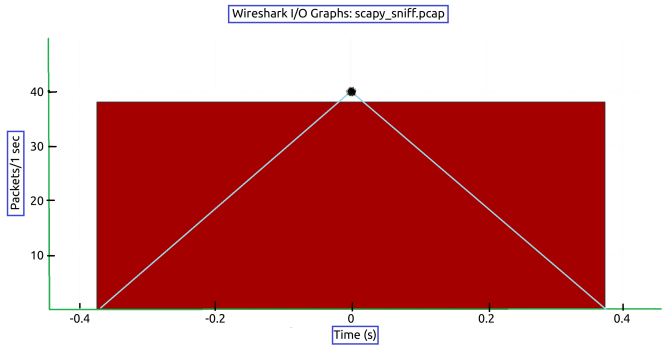


Fig. 16. PCAP Analysis - Scapy IDS - Blacklist Traffic

Attacker's end and also some extra traffic in the same network. From graphs, the illustration shows that Snort detected 2 packets and Scapy has detected 40 packets [Fig 15 & 16]. It is because its count had been set to 40. So after detecting 40 packets, it stopped. Out of which 38 shows TCP error (the red region) because Snort detected them and blocked those packets.

VII. RECONNAISSANCE ON BLACKLISTED IPS

Recapitulating the analysis done before, inferred that some fake packet generated do not belong to conventional internet traffic but possesses standard protocols like TCP, UDP, etc. So, then it raised the need to detect those newly added packets in systems connected under the same LAN.

Now, as a priority to keep the system safe (mitigation technique), utilized the technique of Blacklisting those subnets which can pose a potential threat to all the systems which are connected under the same LAN. Then, proposed some basic reconnaissance over the Blacklisted IPs that were obtained before.

A. *Network Reconnaissance on Blacklists*

Proceeding over the same concept, glanced over some techniques to get some holistic idea of blacklisted IPs. Then, developed a simple python script that would give a plot to get more idea on how those Blacklisted IPs/subnets are connected over to DNS servers.

```
# !/usr/bin/env python3
from scapy.all import *

hosts = ["<host 1>",
"<host 2>", ...]

res,unans = traceroute(hosts)
res.graph(target="> traceroute_graph.svg")
```

Example-1:
Illustrated using the following example and tried to include the Attacker Blacklisted IP address in the *hosts* array. The executed script for the same is as follows:

```
# !/usr/bin/env python3
from scapy.all import *

hosts = ["<dst IP>"]

res,unans = traceroute(hosts)
res.graph(target="> traceroute_graph.svg")
```

Example-2:
The following example was also tried using sub domains of some organisations' domains to get to know about incoming traffic and its sources among various Internet Service Providers and their DNS servers.

```
# !/usr/bin/env python3
from scapy.all import *
```

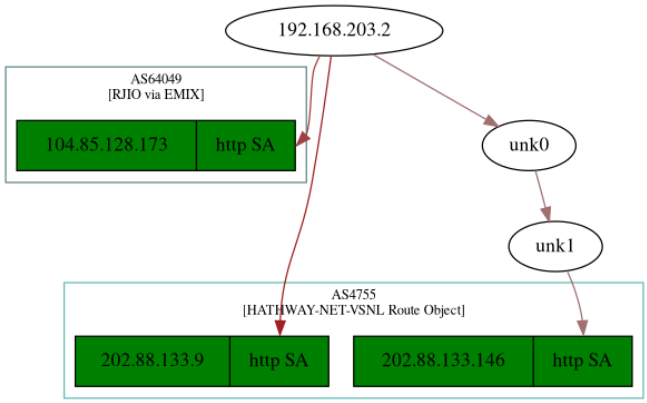


Fig. 17. Graph [Example 1]-Blacklist IP Traffic

```
hosts = ["moodle.iitdh.ac.in",
"iitdh.ac.in", "smp.iitdh.ac.in",
"cdc.iitdh.ac.in", "gitea.iitdh.ac.in"]

res,unans = traceroute(hosts)
res.graph(target="> traceroute_graph.svg")
```

B. *Graph and Inference*

After running the script, defined the graph for Example 1 and Example 2, respectively [Fig 17 & 18].

In Example 1, considered the Attacker's IP Address (`src IP`) as the host and tried to get a basic reconnaissance on it. As evident from Fig 17,that one of the ISPs is Hathway Net and other is RJIO. Also,analysed some of the IP addresses `104.85.128.173 & 202.88.133.9/146` using `traceroute` commands in the scripts. A common methodology of attacker would yield potential threat to the Server's IP addresses exposed in Graph. So, putting some good security measures would help in mitigating the obfuscation of malicious traffic and corresponding potential threats.

In Example 2, analysed using some of the sub domains of the organisation (`.iitdh.ac.in`) as the hosts and tried to get a basic reconnaissance on it. As evident from Fig 18, that one of the ISPs is NKN (National Knowledge Network) and other is BSNL. Also, got some of the IP addresses `14.139.150.68 & 61.0.239.228` using `traceroute` commands in the scripts. A common methodology of attacker would yield potential threat to the Server's IP addresses exposed in Graph. So, putting some good security measures would help in mitigating the obfuscation of malicious traffic and corresponding potential threats.

VIII. Port Security

Nowadays, Ethernet LANs are substantially threatened by tackers because of the leveraging of the security, through some of the default open ports. Address Spoofing and other attacks like DoS (Denial of Service attack), DDoS (Distributed Denial of Service Attack) are notable ones for leading to the penalty for the network system [21]. Thus, the concept of port-security merged wherein the network administrator has control over

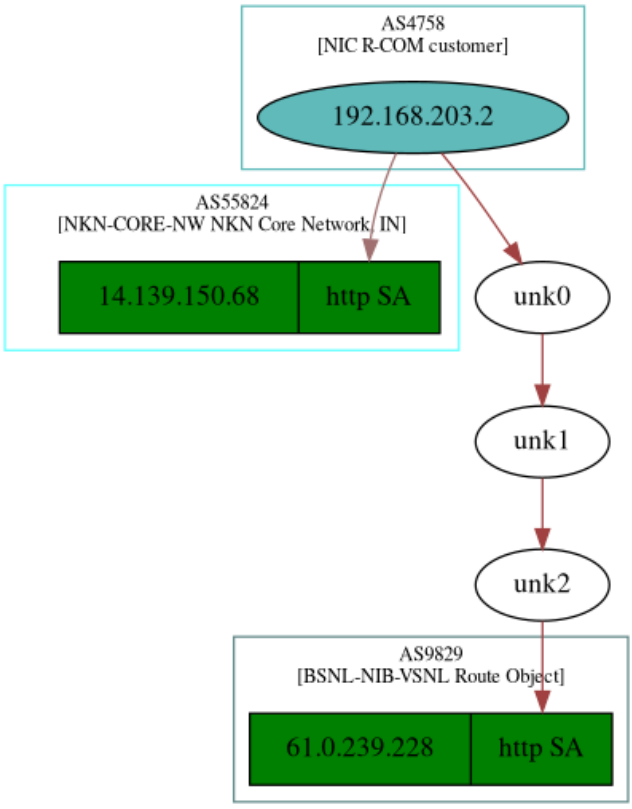


Fig. 18. Graph [Example 2]-Blacklist IP Traffic

the entire schema of the network, its resources, utilities, and users. So, the admin can accordingly choose to control the network. An important understanding for ensuring the security states that the port should be made accessible by the user, for enabling the port function efficiently.

A. *Strategies for ensuring Port-Security*

1) Defining the constraint over the number of MAC addresses permitted over the single switch port. If the limit exceeds the permitted ones, then the appropriate penalty is enforced.
2) Maintaining the logs of the continuous work done by the user helps in monitoring the traffic for keeping an eye on an intruder or unauthorized access to the network.

B. *Violation Modes of Port-Security*

1) **protect-** it enables to leverage of the data packet with some random MAC address until the sufficient secure mac address is lowered underneath the maximum value.
2) **restrict-** An improvement to the protect mode, also helps in defining a log message stream by iteratively summing the counter values, thus sending the SNMP signals.
3) **shutdown-** Improved security in this mode, as it closes the port if suspicious activity is detected.
4) **sticky-** Helps in defining the static mac address instead of manually stating the absolute mac address.

C. *Defined Approach for Port Security*

Port Knocking: Maintaining the monitoring the authorization and accessibility of the port are quite substantial for ensuring port security and a secure connection, too. So, thus the concept of Port knocking is defined which helps in escaping the services available from intruders or attackers by utilizing the closed and inaccessible strongly encrypted closed ports. Defining the authentication framework for the system is also illustrated through the Secure Port Knock-Tunneling (SPKT) by utilizing two phases defined as solving the DOS-knocking attack and removing the NAT-knocking problem [10].

Strategies implemented for ensuring the legal practices are adopted for EU nations, thus ensuring identity control, access management, and defining a proper systematic framework for port facility security plan (PFSP) for defining maximum protection from attacks [9].

IX. DNA Cryptography

The recent advances in the field of DNA computing are quite important for defining an optimal approach for improving efficiency and security, by combing modern biotechnology and cryptology. It can be stated as using a DNA sequence for securing the data. So, basically in DNA cryptology, several DNA algorithms are utilized for analysis namely,' DNASC cryptography system',' DNA Steganography Systems',' Public-key system using DNA as a one-way function for key distribution', 'Triple stage DNA Cryptography', and its optimized related encryption algorithm using the foundation of Chaotic and DNA computing. Encoding the data in the DNA strand is important and is defined using 4 nitrogenous bases as Adenine (A), Thymine (T), Cytosine (C), Guanine (G). Now, one of the simplest adopted approaches for encoding could be A(0)–00, T(1)–01, C(2)–10, G(3)–11. Algorithms for strong encryption and encryption are also illustrated. So, accordingly, users can prefer to store the data in the form of nitrogenous bases, just because it solves the data storage and as well to an extent time complexity issue, too [13].

Various enhanced approaches have also been defined such as defining a new cryptographic technique ensuring parallelism, that utilizes DNA hybridization concept, one-time pad scheme, and DNA molecular structure for limiting and optimizing the time for computation [11][14].

Another approach focused on storing data securely and efficiently on the cloud, by improving security but at the cost of addition of computational complexity as a result of utilizing the bio-computing techniques and for ensuring the data integrity without an involvement for any third-party, though still a lot of improvement in terms of security, cost and storage can be researched, further which will quite beneficial for the future computations [12].

Researchers have illustrated the comparative study on DNA Cryptography using Message transmissions. The DNA sequence is encoded by a Text message. Thus, it reduces the time complexity of the message that is encrypted using a one-time pad and bio-molecular strategies.

X. Quantum Cryptography

Basically, quantum cryptography is utilising the uncertainty principle for its computation. The concept named as Quantum key Distribution (QKD), in which it uses the secret, random and shared sequence of bits for effective communication between two systems or users. Now, after that the information can be exchanged through the different strong cryptographic algorithms [18].

A. *Attacks Approaches in Quantum Cryptography*

1) **Photon Number Splitting (PNS) Attack –** A pulse (consisting of several photons) is sent because it is not feasible to transmit a single photon because of its low intensity, so the intruder can capture and monitor some of the properties and behaviour of photons while transmission and thus use the same polariser as used by the communicating entities for deciphering the key.

2) **Faked State Attack –** In this scenario, the intruder tries to get a replica of one of the communicating entities' photon detector and then captures those photons, modifies the bits in it, and then sends it back to the intended user, thus inculcating the attack.

B. *Related & Optimised works in Quantum Cryptography*

Researchers have defined how quantum cryptography following the contribution of network security has enabled in improving the security by inferring the presence of an intruder or third-party in communication. Now, it depends on the fundamentals of quantum mechanics, which uses Quantum Key Distribution (QKD) techniques for ensuring a highly secure and authenticated communication for real-time usage. As the advancement in the power of the system is increasing rapidly and so is the probability of an attack of is also increasing, which to an extent also caters to the development and contributions towards significant improvements in quantum cryptography [15].

Another work focuses on the implications of quantum cryptography focusing especially on the real-time functioning and efficacy. This again marks the importance of Quantum Key Distribution (QKD) techniques for optimization of algorithms through also catering to providing improved security and preventing unauthorized access, by also improving time and space complexities [16].

Conclusion

The research emphasizes one of the most prevalent MITM(Man-in-the-middle) attacks on Fake Packet Generation and Detection in the network system. It illustrates a versatile and diverse set of possible Fake Packets generated using Scapy Library in Python3. Post various configurations in open-source IDS like Snort, the findings demonstrated the detection of such non-conventional traffic across various systems connected under the same LAN.

Utilizing the mitigation technique, the proposed solutions were helpful to segregate IP addresses/subnets into Whitelist

and Blacklists Traffic. Subsequently, it restricted and secured the systems from receiving potential malicious packets. Also, it catered in avoiding the threat of circulation of malicious packets which in turn are Trojan to Exploits and hence can essentially lead to failure/crash of Computer Network and systems under that LAN. By all counts, and with proven results illustrated through a script that provides some basic information about incoming traffic, DNS server, it's routing, etc. in terms of plot.

Further to ensure optimization, vulnerability assessment and monitoring techniques like Port Security are comprehensively and extensively defined. Various strategies of ensuring security, violations modes, and port knocking are illustrated, which helps in optimizing the time complexity. Some state-of-the-art techniques like DNA security and Quantum security concepts further strengthen the given proposition and can be used to insinuate novel approaches towards network analysis and ensuring security.

REFERENCES

[1] Pansari, N., & Kushwaha, D. (2018). Blended Extensibility of Cyber Forensics. International Journal Of Engineering And Computer Science.,doi: 10.18535/ijecs/v7i4.10

[2] K. F. Kao, W. C. Chen, J. C. Chang and H. T. Chu, "An Accurate Fake Access Point Detection Method Based on Deviation of Beacon Time Interval," 2014 IEEE Eighth International Conference on Software Security and Reliability-Companion, 2014, pp. 1-2, doi: 10.1109/SERE-C.2014.13.

[3] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark," 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN), 2017, pp. 77-81, doi: 10.1109/CICN.2017.8319360.

[4] Pansari, N., & Kushwaha, D. (2018). Advancement in Robust Cyber Attacks-An Overview. International Journal of Research in Engineering, IT and Social Sciences.

[5] Diyeb, I. A. I., Saif, A., & Al-Shaibany, N. A. (2018). Ethical network surveillance using packet sniffing tools: A comparative study. International Journal of Computer Network and Information Security, 11(7), 12.

[6] M. Alajeely, A. Ahmad, R. Doss and V. Mak-Hau, "Packet Faking Attack: A Novel Attack and Detection Mechanism in OppNets," 2014 Tenth International Conference on Computational Intelligence and Security, 2014, pp. 638-642, doi: 10.1109/CIS.2014.15.

[7] H. Kim, H. Lee and H. Lim, "Performance of Packet Analysis between Observer and WireShark," 2020 22nd International Conference on Advanced Communication Technology (ICACT), 2020, pp. 268-271, doi: 10.23919/ICACT48636.2020.9061452.

[8] Shende, O., Pateriya, R.K. & Verma, P. A N-binary Classification and Grouping-based Approach to Improve the Performance of Anomaly Detection. Arab J Sci Eng (2021),doi: 10.1007/s13369-021-05871-6

[9] F. Andritsos, "Port security & access control: A systemic approach," IISA 2013, 2013, pp. 1-8, doi: 10.1109/IISA.2013.6623728.

[10] P. Mehran, E. A. Reza and B. Laleh, "SPKT: Secure Port Knock-Tunneling, an enhanced port security authentication mechanism," 2012 IEEE Symposium on Computers & Informatics (ISCI), 2012, pp. 145-149, doi: 10.1109/ISCI.2012.6222683

[11] Anwar, T., Paul, S., & Singh, S. K. (2014). Message transmission based on DNA cryptography. International Journal of Bio-Science and Bio-Technology, 6(5), 215-222, doi:10.14257/ijbsbt.2014.6.5.22

[12] S. Pramanik and S. K. Setua, "DNA cryptography," 2012 7th International Conference on Electrical and Computer Engineering, 2012, pp. 551-554, doi: 10.1109/ICECE.2012.6471609, doi: 10.1109/ICECE.2012.6471609

[13] Xiao, Guozhen & Lu, Mingxin & Qin, Lei & Lai, Xuejia. (2006). New field of cryptography: DNA cryptography. Chinese Science Bulletin. 51. 1413-1420. 10.1007/s11434-006-2012-5.

[14] Jacob, Grasha & Murugan, Annamalai. (2013). DNA based Cryptography: An Overview and Analysis. International Journal of Emerging Sciences. 3. 36-42.

[15] M. S. Sharbaf, "Quantum Cryptography: A New Generation of Information Technology Security System," 2009 Sixth International Conference on Information Technology: New Generations, 2009, pp. 1644-1648, doi: 10.1109/ITNG.2009.173,doi: 10.1109/THS.2011.6107841.

[16] M. S. Sharbaf, "Quantum cryptography: An emerging technology in network security," 2011 IEEE International Conference on Technologies for Homeland Security (HST), 2011, pp. 13-19, doi: 10.1109/THS.2011.6107841,doi: 10.1109/ITNG.2009.173.

[17] Bruss, D., Erdélyi, G., Meyer, T., Riege, T., & Rothe, J. (2007). Quantum cryptography: A survey. ACM Computing Surveys (CSUR), 39(2), 6-es.

[18] Brassard, G., Lütkenhaus, N., Mor, T., & Sanders, B. C. (2000, May). Security aspects of practical quantum cryptography. In International conference on the theory and applications of cryptographic techniques (pp. 289-299). Springer, Berlin, Heidelberg.

[19] M. Moizuddin, J. Winston and M. Qayyum, "A comprehensive survey: Quantum cryptography," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), 2017, pp. 98-102, doi: 10.1109/Anti-Cybercrime.2017.7905271.

[20] Pansari, N., & Agarwal, A. (2020). A Comparative Study of Analysis and Investigation using Digital Forensics. International Journal of Linguistics and Computational Applications (IJLCA), 7(2).

[21] Gupta, Alka & Sharma, Lalitsen. (2019). Mitigation of DoS and Port Scan Attacks Using Snort. International Journal of Computer Sciences and Engineering. 7. 248-258. 10.26438/ijcse/v7i4.248258.

[22] Bansal, S., & Bansal, N. (2015). Scapy-a python tool for security testing. Journal of Computer Science & Systems Biology, 8(3), 140.

[23] Brahmanand, S. H., Lal, N. D., Sahana, D. S., Nijguna, G. S., & Nayak, P. (2022). A Systematic Approach of Analysing Network Traffic Using Packet Sniffing with Scapy Framework. In Computer Networks and Inventive Communication Technologies (pp. 811-820). Springer, Singapore.

[24] R. Gaddam and M. Nandhini, "An analysis of various snort based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in Kali Linux environment," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), 2017, pp. 10-15, doi: 10.1109/ICICCT.2017.7975177.

[25] Badotra, Sumit & Panda, Surya. (2021). SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking. Cluster Computing. 24. 10.1007/s10586-020-03133-y.