# Implementation, configuration and adaptation of IDS/NSM Linux distribution, IDS Evasion

**MSC in Applied Cybersecurity**

**Assignment – 2**

**Submitted by:**

**Rama Krishna Kambhampaty(B00168197)**

**Lecturer:**

**Mark Lane**

**TECHNOLOGICAL UNIVERSITY OF DUBLIN**

**BLANCHARDSTOWN**

# ABSTRACT

Intrusion detection systems (IDS) have become crucial in ensuring network security. These systems continuously monitor network traffic to detect any unusual activity and alert managers of potential attacks. However, as cyber-attacks become more sophisticated, evasive measures are also growing, making IDS vulnerable to circumvention. This research aims to examine in-depth the various tactics used by malicious actors to evade IDS detection.

The report starts by laying a solid foundation for IDS concepts. It covers their fundamental role, functions, and limitations which can inadvertently lead to loopholes in evasion. This understanding is essential for studying the intricate technique of IDS evasion, which includes packet obfuscation, fragmentation, tape overlapping, protocol manipulation, tunneling, transport insertion, and advanced circumvention techniques.

The research then shifts its focus to practical application by demonstrating the effectiveness of various evasion tactics against open-source IDS platforms such as Snort or Bro. The graphical representation of these strategies' ability to evade IDS detection in a hands-on exercise demonstrates the practical challenges faced by Integrated Defense Systems against new threats, such as maintaining their effectiveness.

The research aims to improve IDS detection abilities to deal with persistent evasion attempts. Implementing defense and analysis tools, as well as updates to signatures and anomaly detection algorithms, could enable IDS to combat advanced evasion strategies and remain a solid security precaution.

# INTRODUCTION

## Intrusion Detection System

Intrusion Detection Systems (IDS) are essential tools in safeguarding networks and systems from malicious attacks. These systems monitor network traffic and system activities for signs of suspicious or anomalous behavior. Identifying potential threats early on empowers network administrators to take proactive measures to neutralize attacks and protect sensitive data. IDS have become an integral component of any comprehensive cybersecurity strategy, serving as the first line of defense against a relentless array of cyber threats.[1]

## Types of Intrusion Detection System:

There are two primary types of IDS based on their deployment and monitoring scope:

1. **Host Based Intrusion Detection System:**

   Host-based Intrusion Detection Systems (HIDS) are security software that are installed directly on individual servers or workstations. These systems monitor the activity on the host-level, including system logs, file changes, and process behavior. HIDS are highly effective in detecting unauthorized access, data alterations, and suspicious processes running on the host.

2. **Network Based Intrusion Detection System:**

   Network Intrusion Detection Systems (NIDS) are usually installed on network equipment such as routers or firewalls. They constantly monitor incoming and outgoing network traffic and scrutinize network packets for specific patterns associated with known attacks. For example, they can identify SYN floods or SQL injection attempts. This method provides a comprehensive understanding of network activity and helps to detect potential security threats in a holistic manner.

## IDS Detection Techniques

IDS employ various detection methods to identify intrusions:

1. **Signature-based Detection:** This is the most common technique, utilizing a database of known attack signatures. When network packets or host-level events match these

signatures, an alert is triggered, alerting administrators of potential attacks. This method is efficient in detecting known threats but may miss newly devised attacks.

2. **Anomaly-based Detection:** This approach analyzes network traffic or system activity patterns, identifying deviations from normal behavior. Unusual spikes in network traffic, sudden changes in system configurations, or unexpected file access patterns can indicate potential intrusions. Anomaly-based detection is more effective against unknown or zero-day attacks but may generate false positives.

3. **Statistical Anomaly Detection:** A more advanced variation of anomaly-based detection, statistical anomaly detection utilizes statistical methods to identify deviations from established patterns. It relies on probability models to distinguish between legitimate and suspicious activity, offering greater accuracy and reducing false positives.

# METHODOLOGY

This section discusses about the configuration and setup of Intrusion Detection System(IDS) and the different virtual machines or tools that are used in performing the respective process.

## A. SNORT:

In this project, we used snort as the Intrusion detection tool. Here, we installed snort in the ubuntu virtual machine. We used the kali virtual machine as the attacking machine and ubuntu as the target machine.

Snort is a lightweight, modular, and extensible framework capable of detecting and classifying network traffic based on a variety of criteria. It utilizes rule-based detection, where administrators define patterns or signatures that indicate potential intrusions. When Snort encounters traffic matching these rules, it generates alerts, notifying network administrators of potential threats.[2]

Snort is a highly popular and effective intrusion detection and prevention system that offers a wide range of features. These features include real-time traffic analysis, protocol analysis, content searching and matching, and flexible rule-based detection. With its ability to detect a variety of threats, such as viruses, malware, and unauthorized access attempts, Snort has become a go-to solution for network security professionals. Additionally, Snort is an open-source tool that is constantly being updated by a large community of developers, ensuring that it remains a reliable and effective solution for years to come.

There are other tools available for intrusion detection system such as Bro and Suricata, where they support both the signature based and anomaly based detection, offering greater flexibility in the intrusion detection.

## B. Evasion Techniques:

### 1. Packet Fragmentation:

Packet fragmentation divides large network packets into smaller segments for transmission over networks with size limitations. However, attackers can also use it as an IDS evasion technique by breaking malicious packets into smaller segments, making it difficult for IDS to identify the attack. IDS rely on signature-based detection, and fragmented traffic may not match the signatures.[4]

2. **Payload Obfuscation:**

   Payload obfuscation is a cybersecurity technique that obscures or modifies malicious code to evade detection by security systems. Attackers use encoding, encryption, or obfuscation to bypass signature-based detection and hinder analysis. This technique is crucial in advanced evasion methods to enhance malware stealth, bypass intrusion detection systems and antivirus solutions. Payload obfuscation is a key consideration in developing and detecting sophisticated cyber threats.

3. **Packet Overlapping:**

   Packet overlapping is a cybersecurity technique used during network-based attacks. Attackers manipulate packet fragmentation to exploit system vulnerabilities and confuse security devices. This technique exploits weaknesses in systems that handle fragmented packets, making it difficult for defenders to identify and analyze malicious activities. It is part of a broader range of evasion tactics that highlights the ongoing battle between attackers and defenders in the ever-evolving landscape of network security.

4. **Protocol Manupulation:**

   Protocol manipulation in cybersecurity involves exploiting vulnerabilities by manipulating network communication protocols. By sending unexpected protocol data, attackers can deceive or overwhelm security mechanisms, leading to unauthorized access, data exfiltration, or denial-of-service attacks. As defenders fortify protocols against manipulation, adversaries evolve their tactics, making cybersecurity a perpetual battleground.

5. **Tunneling:**

   Tunneling is a network technique to encapsulate data packets of one protocol within another protocol's payload. This allows data to traverse networks or systems where it may not be allowed due to network restrictions or security measures. Tunneling creates a secure and private communication channel and is used to bypass firewalls, establish VPNs, and enable secure communications over public networks. However, it can also be misused by malicious actors. Understanding tunneling is crucial to enhance network security and recognize potential cybersecurity threats.

6. **Traffic Insertion:**

   Traffic insertion is a technique used by attackers to inject malicious data into network traffic. It can be used to introduce malicious content, commands, or activities into ongoing network communications. Attackers use unauthorized traffic to exploit vulnerabilities, compromise systems, or facilitate various cyber threats. They may leverage this technique at different network layers to evade detection by intrusion detection systems or manipulate communication protocols. Network security professionals must understand and mitigate the risks associated with traffic insertion to safeguard against cyber threats that exploit weaknesses in data transmission.

## IDS Evasion Tools:

1. **Metasploit:**

   Metasploit is an open-source framework for penetration testing, developed by Rapid7. It has a modular framework with pre-built exploits, payloads, and auxiliary modules, and supports command-line and graphical interfaces. Metasploit simplifies identifying and exploiting vulnerabilities in various applications and OS. It provides post-exploitation tools, including Meterpreter payload, that allow users to maintain access, gather information, and execute commands on compromised systems. However, it's crucial to use Metasploit responsibly and ethically with proper authorization.

2. **Ptunnel:**

   Ptunnel, which stands for "Ping Tunnel," is a network tool that allows users to bypass firewalls or network restrictions by encapsulating TCP connections within ICMP (Internet Control Message Protocol) packets. This tool was developed for Unix-like systems, including Linux, and is particularly useful for establishing connections across networks where other forms of direct communication may be restricted. By disguising TCP traffic as ICMP echo requests and replies, Ptunnel enables users to create covert communication channels. However, it's important to note that Ptunnel should be used ethically and legally, with proper authorization, as it has the potential to be misused. While it can serve legitimate purposes, such as accessing services through firewalls, users must exercise caution and follow guidelines to avoid any ethical or legal issues.

3. **Scapy:**

Scapy is an open-source Python-based tool and library for network analysis and penetration testing. It provides a flexible interface for constructing, sending, and capturing network packets, enabling users to create custom network protocols, perform various network-related tasks, and analyze network traffic. Scapy is capable of packet sniffing, forging, and decoding, making it a valuable tool for network reconnaissance, debugging, and security assessments. It supports a wide range of protocols and is used by network administrators, security professionals, and ethical hackers to gain insights into network behavior and test the security posture of systems. Scapy's interactive shell and scripting capabilities make it a preferred choice for those seeking a dynamic and adaptable tool in the realm of network exploration and analysis.

4. **Nmap:**

Nmap is a network scanning tool used to gather information about network architecture. Though it is not designed for malicious activities, attackers may use it to identify potential network vulnerabilities. Nmap allows users to perform host discovery, port scanning, and service version detection. However, it is important to use Nmap responsibly and ethically within legal boundaries and with proper authorization. Its usage may trigger alerts from Intrusion Detection Systems due to its network-scanning nature. Therefore, it is crucial to ensure that Nmap is used for network reconnaissance and assessment purposes only.

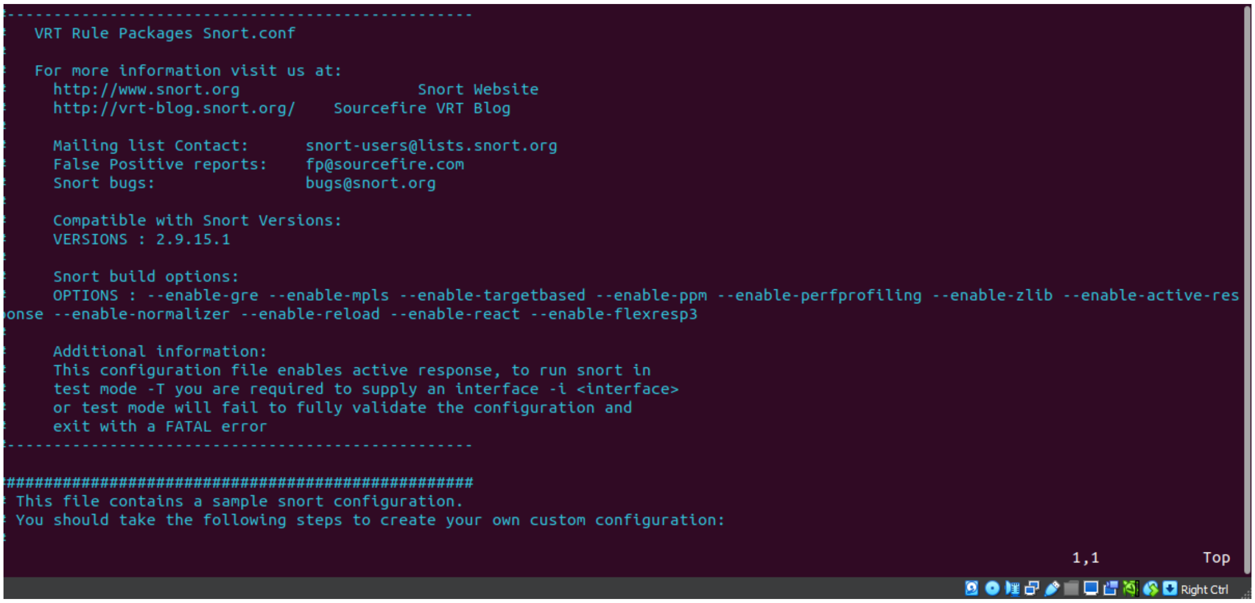
# IMPLEMENTATION

## Installing and Configuring Snort:

In this project, we installed snort in the ubuntu virtual machine. To install snort, we use command “sudo apt install snort -y” and during the installation we also specify the address range which in this case 192.168.0.0/24. We should set the promiscuous mode to Allow All for the successful snort configuration.[5]

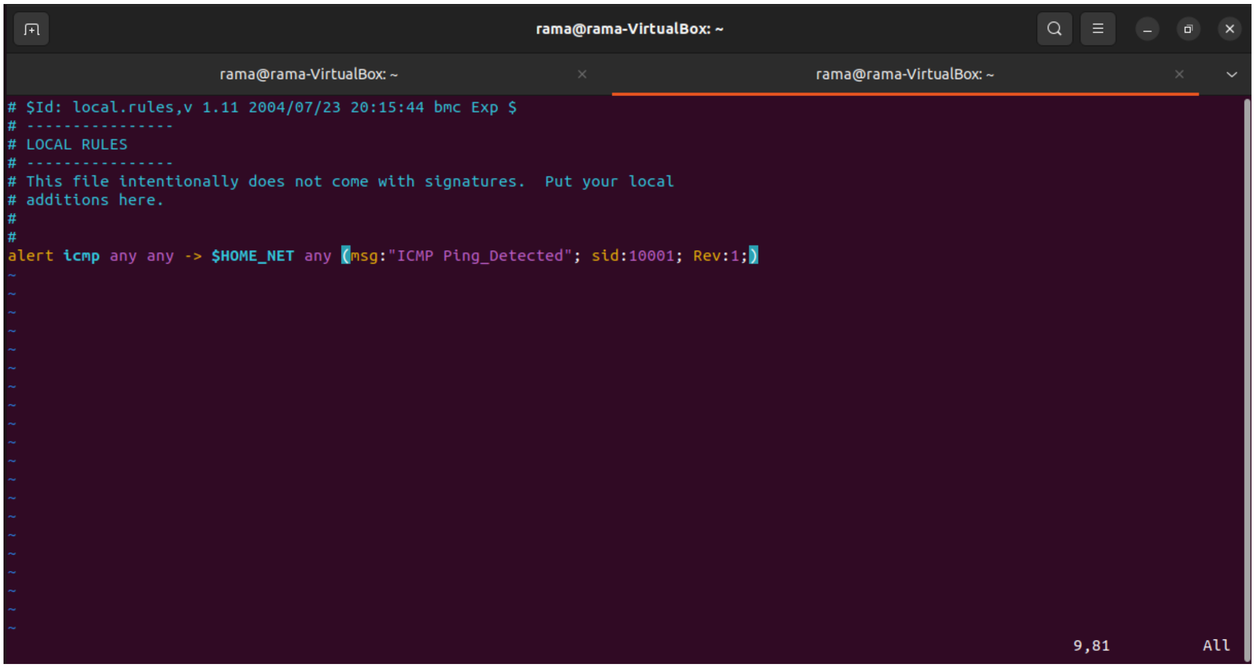
We can check the snort version using command snort –version, and to configure the snort we use the command “sudo vim /etc/snort/snort.conf”. Snort.conf file is core of the snort functionality. We can configure rules or enable and disable the snort rules with this command.

During snort configuration, at the step1 we set the network variables. Here, we set the $HOME_NET to our specified IP address 192.168.0.0/24 and we leave the &EXTERNAL_NET to ‘any”.



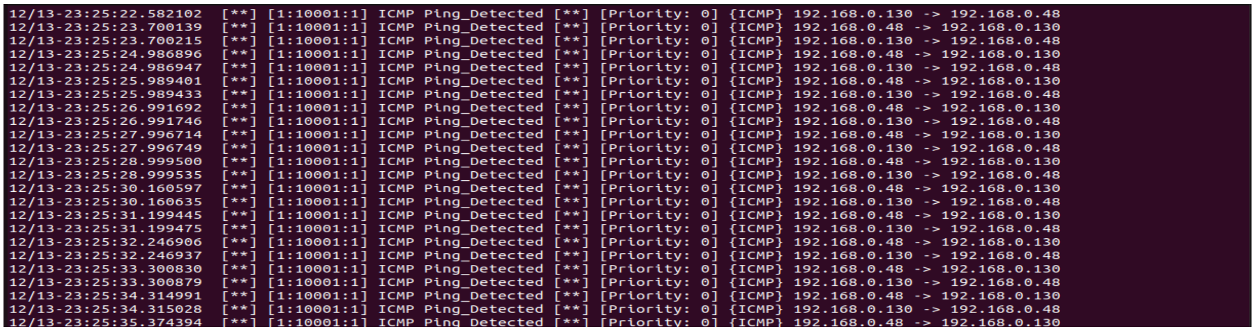
```
--------------------------------------------------
   VRT Rule Packages Snort.conf

   For more information visit us at:
     http://www.snort.org                   Snort Website
     http://vrt-blog.snort.org/    Sourcefire VRT Blog

     Mailing list Contact:      snort-users@lists.snort.org
     False Positive reports:    fp@sourcefire.com
     Snort bugs:                bugs@snort.org

     Compatible with Snort Versions:
     VERSIONS : 2.9.15.1

     Snort build options:
     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --enable-active-res
onse --enable-normalizer --enable-reload --enable-react --enable-flexresp3

     Additional information:
     This configuration file enables active response, to run snort in
     test mode -T you are required to supply an interface -i <interface>
     or test mode will fail to fully validate the configuration and
     exit with a FATAL error
--------------------------------------------------

##################################################
 This file contains a sample snort configuration.
 You should take the following steps to create your own custom configuration:

                                                                                                                   1,1           Top
```

To configure our rules in local rules we use command “sudo vim /etc/snort/rules/local.rules”



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
#
#
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping_Detected"; sid:10001; Rev:1;)
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                              9,81          All
```

This rules generates alerts whenever a ping was initiated.



```
12/13-23:25:22.582102  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:23.700139  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:23.700215  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:24.986896  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:24.986947  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:25.989401  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:25.989433  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:26.991692  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:26.991746  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:27.996714  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:27.996749  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:28.999500  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:28.999535  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:30.160597  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:30.160635  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:31.199445  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:31.199475  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:32.246906  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:32.246937  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:33.300830  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:33.300879  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:34.314991  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
12/13-23:25:34.315028  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.130 -> 192.168.0.48
12/13-23:25:35.374394  [**] [1:10001:1] ICMP Ping_Detected [**] [Priority: 0] {ICMP} 192.168.0.48 -> 192.168.0.130
```

Similarly,  to detect the SSH Authentication alerts we configure the rule as:

“alert tcp any any -> $HOME_NET 22 (msg:”SSH Authentication Attempt Detected”; sid:10001; Rev:1;)”

This rule detects if any SSH connection and generates the alert message.

We can create and configure our own rules depending on our requirements.  The stronger the rule is the better snort will be able to detect the attempts.

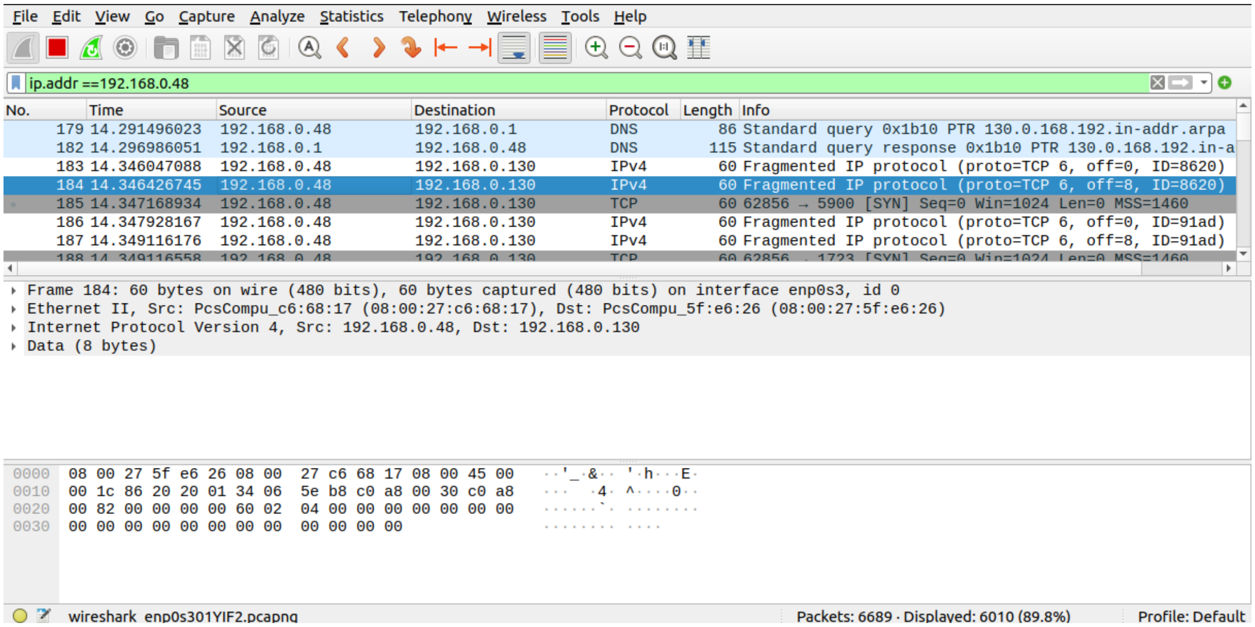
**Implementation of IDS Evasion Techniques:**

We discussed about the various types of Evasion techniques in the above sections. Here we used the nmap tool to evade the snort IDS.

The first method we used is the “Packet fragmentation”.[5][6]

1. **Implementing Packet fragmentation with nmap:**

   This option evade pattern matching detection technique. Since packet reassembly can be quite processor intensive, it's common for admin to disable it. In snort, fragmentation reassembly functionality is disabled by default.
   **Usage:** nmap -f 192.168.0.130



```
ip.addr ==192.168.0.48
```

```
No.     Time           Source          Destination      Protocol  Length Info
    179 14.291496023   192.168.0.48    192.168.0.1      DNS         86 Standard query 0x1b10 PTR 130.0.168.192.in-addr.arpa
    182 14.296986051   192.168.0.1     192.168.0.48     DNS        115 Standard query response 0x1b10 PTR 130.0.168.192.in-a
    183 14.346047088   192.168.0.48    192.168.0.130    IPv4        60 Fragmented IP protocol (proto=TCP 6, off=0, ID=8620)
    184 14.346426745   192.168.0.48    192.168.0.130    IPv4        60 Fragmented IP protocol (proto=TCP 6, off=8, ID=8620)
    185 14.347168934   192.168.0.48    192.168.0.130    TCP         60 62856 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
    186 14.347928167   192.168.0.48    192.168.0.130    IPv4        60 Fragmented IP protocol (proto=TCP 6, off=0, ID=91ad)
    187 14.349116176   192.168.0.48    192.168.0.130    IPv4        60 Fragmented IP protocol (proto=TCP 6, off=8, ID=91ad)
    188 14.349116558   192.168.0.48    192.168.0.130    TCP         60 62856 → 1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

```
▸ Frame 184: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_c6:68:17 (08:00:27:c6:68:17), Dst: PcsCompu_5f:e6:26 (08:00:27:5f:e6:26)
▸ Internet Protocol Version 4, Src: 192.168.0.48, Dst: 192.168.0.130
▸ Data (8 bytes)
```

```
0000  08 00 27 5f e6 26 08 00  27 c6 68 17 08 00 45 00   ··'_·&·· '·h···E·
0010  00 1c 86 20 20 01 34 06  5e b8 c0 a8 00 30 c0 a8   ···  ·4· ^····0··
0020  00 82 00 00 00 00 60 02  04 00 00 00 00 00 00 00   ······`· ········
0030  00 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

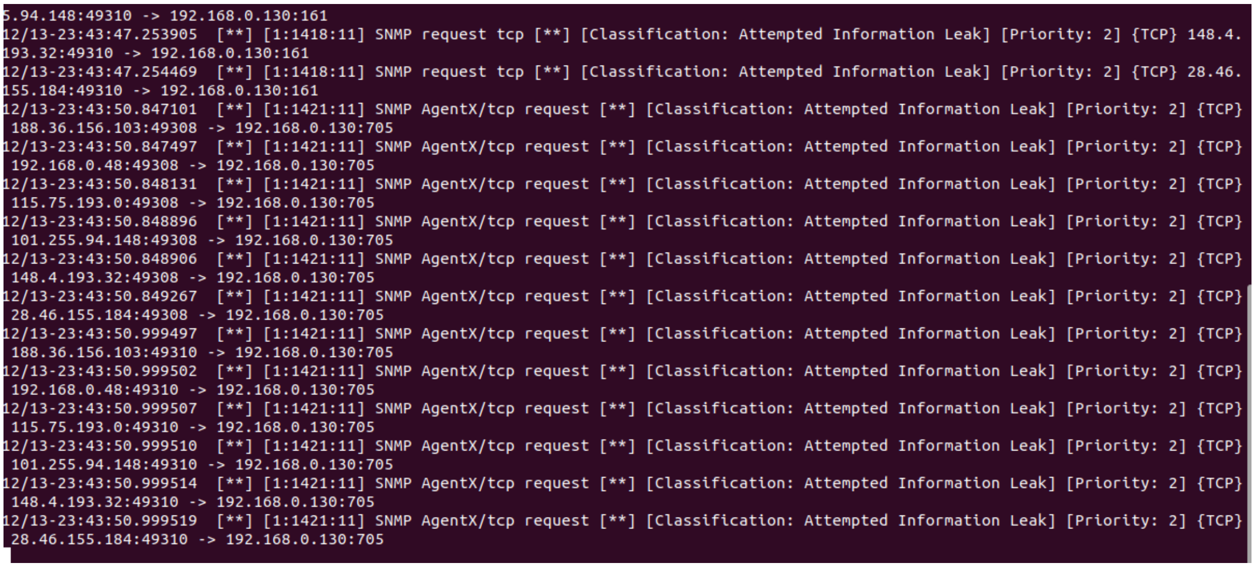
```
wireshark_enp0s301YIF2.pcapng          Packets: 6689 · Displayed: 6010 (89.8%)          Profile: Default
```

   Here we can see the data 8 bytes which is the fragmented data with the use of -f.

2. **Decoy Scan:**

   We can add some random hosts either from the attacker's subnet or from victim's subnet while scanning the target. In firewall logs, there will be multiple hosts along with the attacker's IP making it difficult to trace the attacker.

   **Usage:** nmap -D



```
5.94.148:49310 -> 192.168.0.130:161
12/13-23:43:47.253905  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 148.4.
193.32:49310 -> 192.168.0.130:161
12/13-23:43:47.254469  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 28.46.
155.184:49310 -> 192.168.0.130:161
12/13-23:43:50.847101  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 188.36.156.103:49308 -> 192.168.0.130:705
12/13-23:43:50.847497  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 192.168.0.48:49308 -> 192.168.0.130:705
12/13-23:43:50.848131  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 115.75.193.0:49308 -> 192.168.0.130:705
12/13-23:43:50.848896  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 101.255.94.148:49308 -> 192.168.0.130:705
12/13-23:43:50.848906  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 148.4.193.32:49308 -> 192.168.0.130:705
12/13-23:43:50.849267  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 28.46.155.184:49308 -> 192.168.0.130:705
12/13-23:43:50.999497  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 188.36.156.103:49310 -> 192.168.0.130:705
12/13-23:43:50.999502  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 192.168.0.48:49310 -> 192.168.0.130:705
12/13-23:43:50.999507  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 115.75.193.0:49310 -> 192.168.0.130:705
12/13-23:43:50.999510  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 101.255.94.148:49310 -> 192.168.0.130:705
12/13-23:43:50.999514  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 148.4.193.32:49310 -> 192.168.0.130:705
12/13-23:43:50.999519  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
 28.46.155.184:49310 -> 192.168.0.130:705
```

Here for the decoy we assigned RND: 5 which generates 5 more decoy ip address along with our source ip which makes it harder for the ids to detect.

3. **Spoof source IP address:**

   Attacker can spoof the source IP address (from the victim's subnet) so that it'll appear to IDS/firewall that it's legitimate user and will be passed.

   **Usage:**nmap -S

## Implementation with Metasploit:

The Metasploit tool is used to send exploits by examining the vulnerability of the target system. Here we use the following exploit to evade the ids:

Msf6> exploit(unix/ftp/’vsftpd_234_backdoor)> set rhosts 192.168.0.130 #we give target ip

Msf6> exploit(unix/ftp/’vsftpd_234_backdoor)> set lhost 192.168.0.48 #we give our ip

Msf6> exploit(unix/ftp/’vsftpd_234_backdoor)> set

# CONCLUSION

Finally, this IDS research has given us important insight into the intricacies of intrusion detection systems and the different evasion strategies that might undermine their efficiency. We obtained a better knowledge of the weaknesses in IDS by studying and developing evasion strategies such as payload obfuscation, packet fragmentation, packet overlapping, protocol manipulation, tunneling, and traffic insertion. The actual use of these approaches against the open-source intrusion detection system Snort (or Bro) revealed possible flaws in existing settings.

Furthermore, the project's simultaneous emphasis on offensive and defense highlighted the significance of a complete strategy to cybersecurity. While evasion strategies are dangerous, the study underscored the need of using defensive and analytic technologies to identify, neutralize, and guard against advanced assaults. The inclusion of lab tools, together with IDS training to spot evasion strategies, adds to the overall resilience of the system.

As the cybersecurity landscape evolves, this research underscores the dynamic nature of threats and the ongoing need for robust defense mechanisms. The findings not only enhance our understanding of IDS vulnerabilities but also provide practical recommendations for fortifying these critical components of network security. Moving forward, continual research and adaptation will be essential to staying ahead of evolving cyber threats and ensuring the resilience of intrusion detection systems in increasingly sophisticated digital environments.

# REFERENCES

1. https://infosecwriteups.com/evading-firewall-ids-during-network-reconnaissance-using-nmap-7dc393138178
2. Intrusion detection system: A comprehensive review Author links open overlay panelHung-Jen Liao a, Chun-Hung Richard Lin a, Ying-Chih Lin a b, Kuang-Yuan Tung a
3. Survey of intrusion detection systems: techniques, datasets and challenges Published: 17 July 2019 volume 2, Article number: 20 (2019)
4. https://security.stackexchange.com/questions/121900/how-can-the-nmap-tool-be-used-to-evade-a-firewall-ids
5. Nmap documentation https://nmap.org/book/man-bypass-firewalls-ids.html
6. https://www.youtube.com/watch?v=DSRCx1RpxIg
7. Network Intrusion Detection Systems Evasion Techniques – an Investigation Using Snort J.A.Ytreberg and M.Papadaki