JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

MiTM Using Custom Packet Sniffer

Submitted in partial fulfillment of the requirements of the degree of bachelor's in engineering by

Devina Panchal	SE-15 Roll no. 33
Yogeshchand Rai	SE-15 Roll no. 41
Swaraj Sakpal	SE-15 Roll no. 44
Vansh Visaria	SE-15 Roll no. 59

Under the guidance of

Ms. Pranali Pawar

Dr. Asha Durafe

Guide:

1. Dr. Asha Durafe
2. Ms. Pranali Pawar

Chapter 1

1.1 Introduction

To intercept network packets within the same network, we created a specialised packet sniffer program. The purpose of this program, which makes use of the Python scapy module, is to assist network engineers and users interested in keeping an eye on their network traffic. An advanced port scanning technology that identifies open ports connected to a particular IP address is a distinguishing feature. Our main objective is to create a unique packet sniffer using the Python 3 scapy library, which will enable us to monitor and control the flow of packets across a network.1.2

1.2 Purpose of the Project

The internet connects everything in the modern world through a variety of networks,

some of which are open to the public (like the WiFi at Starbucks restaurants and cafes) and other of which are private (like the WiFi at home).
This suggests that there is a chance that everyone else on the same network could see what you do online.

We decided to create a tool to monitor the network that is sniffing out the packets because of this. Although we are aware that this tool may seem disagreeable, we believe that network specialists and anyone else who wishes to monitor their own network traffic will find it to be of great use.
Because it is relatively simple and lightweight, this program is perfect for beginners to learn networking concepts much more rapidly.

1.3 Need for Project

Finding simple, beginner-friendly tools for studying networking fundamentals like packet sniffing, port scanning, MITM, etc. can be challenging.

Therefore, we decide to use Python to develop a unique tool that can meet the aforementioned requirements.

We made sure that this tool was easy to set up and that the UI was simple to understand so that people could learn about the ideas rather than struggle with the application.

1.4 Objectives

The objectives of our project include sniffing the packets that enter and depart the device it is running on and monitoring or modifying them.

Additionally, we want to make sure that this project is very user-friendly, so we'll upload it to GitHub so that anyone may inspect the code and utilise it on their computers.

Additionally, we made an effort to make it as light as possible so that using this packet sniffer application doesn't actually depend on the hardware configuration of on computer.

Chapter 2

Review Of Literature

[1] The author of the paper "Port scanning detection based on anomalies" discusses the features of network traffic and how their values affect the network's performance. An anomaly or a breach in network performance is indicated by a large change in their values. The types of network anomalies created by port scans are discussed in this article.

[2] The author of the paper "Slow port scanning detection" discusses intrusion detection, a tool used to identify various network attacks on wired or wireless networks. One of the risky attacks that intrusion detection looks for is port scanning. Stanford University researchers have created a novel method for detecting port scanning attacks that is based on fuzzy logic.

[3] The article "Penetration Testing Active Reconnaissance Phase - Optimised Port Scanning With Nmap Tool" discusses how to manage traffic accountability and the amount of time needed to finish a given task during the reconnaissance phase of active scanning with the Nmap tool. The black hat uses methods for passively obtaining information. The attacker begins scanning the perimeter and internal network devices after they have amassed enough statistics.

[4] The author of the paper "On Detecting Port Scanning using Fuzzy Based Intrusion Detection System" discusses intrusion detection, a tool for identifying various network attacks on wired or wireless networks. One of the risky attacks that intrusion detection looks for is port scanning. Stanford University researchers have created a novel method for detecting port scanning attacks that is based on fuzzy logic.

[5] The author discusses port knocking, a potentially useful tool that has a number of weaknesses including TCP replay, port scanning, and others, in this work titled "Simple Port Knocking Method: Against TCP Replay Attack and Port Scanning". By utilising the Source Port sequences, this research suggests an alternative strategy to the current Port Knocking. The operating system automatically creates the Source port, which is pre-assigned to create a sequence.

[6] The author of the essay "Network forensic system for port scanning attack" discusses network forensics, which is watching network traffic and figuring out whether an anomaly in the traffic points to an attack. Investigators can identify and bring charges against the attackers thanks to network forensic technology. In order to handle enormous amounts of network data, this study suggests a straightforward design for network forensics. It gathers and stores the data using open source network security tools.

[7] The author discusses port scanning in this study, "Probe Delay Based Adaptive Port Scanning for IoT Devices with Private IP Address Behind NAT," which produced a large number of probe and answer packets and could lead to frequent congestion and heavy traffic loads. In order to balance network performance and security requirements for the IoT, the study proposes a probe delay based adaptive scanning algorithm called ProDASA. ProDASA adaptively modifies port scanning frequency and scanning methodologies.

[8] The author discusses the Man-In-The-Middle (MITM) assault, one of the main methods used in computer-based hacking, in this essay titled "Different flavours of Man-In-The-Middle attacks, consequences, and practicable solutions." assaults like Denial of Service (DoS), DNS spoofing, and Port theft can be successfully triggered by MITM assaults. This study focuses on several MITM attack types, their effects, and workable responses.

[9] An article titled "Man-in-the-middle attack on BB84 protocol and its defence" discusses a man-in-the-middle attack on the BB84 protocol. The man in the middle has the ability to send his own message to the recipient after intercepting the communication from the sender. On various quantum key distribution techniques, the attack can be applied analogously. Some strategies for defence against the attack are offered.

[10] The Internet of Medical Things is vulnerable to Man-in-the-Middle (MitM) attacks, as discussed in the paper "Man-in-the-Middle Attack Mitigation in the Internet of Medical Things" by the author. It can recognise medical emergencies in patients who are being watched and repeat regular physiological data to stop the device from sounding an alarm. With a message authentication code, their framework sends a smaller-sized signature that is derived from obtained data.

[11] The author discusses the traditional optical channel's Poissonian statistical behaviour in this study, "Physical Layer Security Protocol for Poisson Channels for Passive Man-in-the-Middle Attack," and proposes a novel secrecy coding-based physical layer protocol. Their system uses the Poisson channel's randomness to secure conventional digital information bits at the photonic level. They introduce a two-way protocol that, regardless of the circumstances of legitimate and eavesdropper channels, always ensures private communication.

[12] The author discusses a man-in-the-middle assault on 3G-WLAN interworking, one of the extensively deployed technologies, in an essay titled "A Man-in-the-Middle Attack on 3G-WLAN Interworking". A gateway is required for protocol shifting since WLAN uses PS (Packet Switch) whereas 3G uses Circuit Switch. During the protocol transformation and codec transcoding, the attacker can listen in on all 3G or WLAN traffic.

[13] The author discusses security, which has turned into a significant barrier for web applications, in this work titled "A Study of Man-in-the-Middle Attack Based on SSL

Certificate Interaction". The user's attention shifts to how to ensure the transfer of sensitive information effectively. Secure Sockets Layer (SSL) evolves in this situation as the circumstances demand. Based on an analysis of the protocol implementation mechanism, we investigate the SSL Handshake Protocol implementation process.

[14] The article "Modelling of Man-in-the-Middle Attack in the Wireless Networks" This essay taught us that Middle assaults are a serious threat to the security of wireless networks. A unified mathematical model is developed in this research to analyse these threats in various wireless networks. A system's susceptibility to this kind of assault can be determined using the model and the logical reasoning employed in it.

[15] The author of the paper "Malicious sniffing systems detection platform" discusses sniffers, which are applications that enable a host to collect any Ethernet network packets. Many fundamental services, including SMTP and FTP, transmit data and passwords in clear text within packets. Hackers can employ sniffers to collect passwords and private information.

[16] The author of the paper "Improving the Analysis of Lawfully Intercepted Network Packet Data Captured for Forensic Analysis" discusses packet sniffers, which are tools used to record all network packets going to and from a suspect's computer. The utility of current packet sniffer and protocol analysis programmes for conducting criminal investigations is constrained. The Highly Extensible Network Packet Analysis (HENPA) framework, which analyses the data to retrieve potentially forensic information, is described in this study.

[17] The experiments were done using the Wireshark packet sniffer to write captured packets straight to disc, as described in the publication "Bottleneck Analysis of Traffic Monitoring Using Wireshark" by authors A. Dabir and A. Matrawy. The findings demonstrated that boosting buffering at the kernel or application levels can greatly enhance capture performance. Combining more kernel socket buffering with a multithreaded capture programme will produce the greatest results.
[18] The author of the paper "ARP spoofing detection algorithm using ICMP protocol" discusses the ARP spoofing attack, which is one of the simplest but most destructive techniques in local area networks. In order to identify malicious hosts, this study suggests an effective method based on the ICMP protocol. It can detect the true address mappings during an attack and won't interfere with host activity on the network.

[19] The author of the paper "Visual Learning Tools for Teaching/Learning Computer Networks: Cisco Networking Academy and Packet Tracer" discusses how a method known as visual learning can help teach and learn more abstract subjects like computer networks. When students can see something in action, they may more readily comprehend and apply the information from the subject. This article describes Packet Tracer, a visual learning tool and network simulator, and its features.

[20] Malicious users deploy a variety of assaults at various levels to steal various levels of data in this document, "A survey on sniffing attacks on computer networks." DHCP Attacks, DHCP Starvation Attacks, and Address Resolution Protocol (ARP) Spoofing are a few sniffing techniques that can be applied at various networking/transmission levels.

Chapter 3

COMPARATIVE ANALYSIS

Sr. No.	Terms	MiTM Packet Sniffer	Other Tools
1.	**Weight of Script**	Light weight Program for Packet Sniffing	Heavy Weight Programs
2.	**Tools Used**	Scapy Library	Tcpdump, Wireshark, etc.
3.	**User Interface**	No User interface	Graphical User Interface is present
4.	**Complexity**	Easy to use and Sniff Packets	Complex Interface for Sniffing
5.	**Task**	Not much Task to Perform	Different actions available for Manipulation of Packets
6.	**Protocols Filtered**	HTTP, TCP, IP, ARP, UDP	HTTP, TCP, IP, etc.
7.	**No. of Tasks that can be Performed**	1) Packet Sniffing 2) Port Scanning 3) Filtering Protocol Filters 4) Viewing Packet Data	1) Packet Sniffing 2) Different Protocol Filters 3) Packet Printing 4) Viewing Data
8.	**Use Cases**	Can be used for Sniffing Packets, Viewing Packet Data, Scanning numbers of Ports	Can be used for Packet Sniffing, Analyzing Data Packets, Manage Protocols, Knowing Network Traffics
9.	**Limitations**	Cannot Scan Packets with HTTPS Protocol	Cannot Scan Packets with HTTPS Protocols

Chapter 4

Implementation

4.1 Applications Used

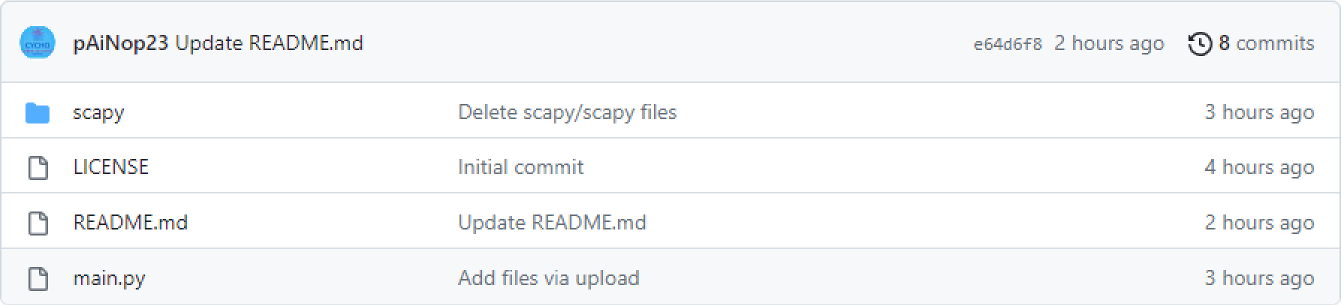
1. Pycharm: An IDE for Python that was used for extensive programming
2. VS Code: It was used for customization and debugging support
3. Notepad for making minor and quick changes
4. Cmder shell in VS Code was used to test and run the code

1) Library

This script was written in Python using the SCAPY package, a powerful interactive Python-based packet manipulation tool. It may do a wide range of tasks, including sending packets across the wire, capturing them, saving or reading them using PCap files, matching requests with answers, forging or decoding packets of various protocols, and more. It aims to facilitate speedy packet prototyping by using functional default settings.

2) Setup

Python is now almost always pre-installed on devices, but if it isn't, you may get it from www.python.org. The software cannot be executed without Python. The next step is to install the scapy library, which may be done by going to www.pypi.org/project/scapy/.Although your environment is now ready for code execution, the code itself is still missing. The code is available at https://github.com/pAiNop23/Packet-Sniffer-Port-Scanner.git on the website.The most effective way to distribute code is through this active github repository. Clone the repository, use the python3 command to launch the "main.py" file from your terminal, and you're done.

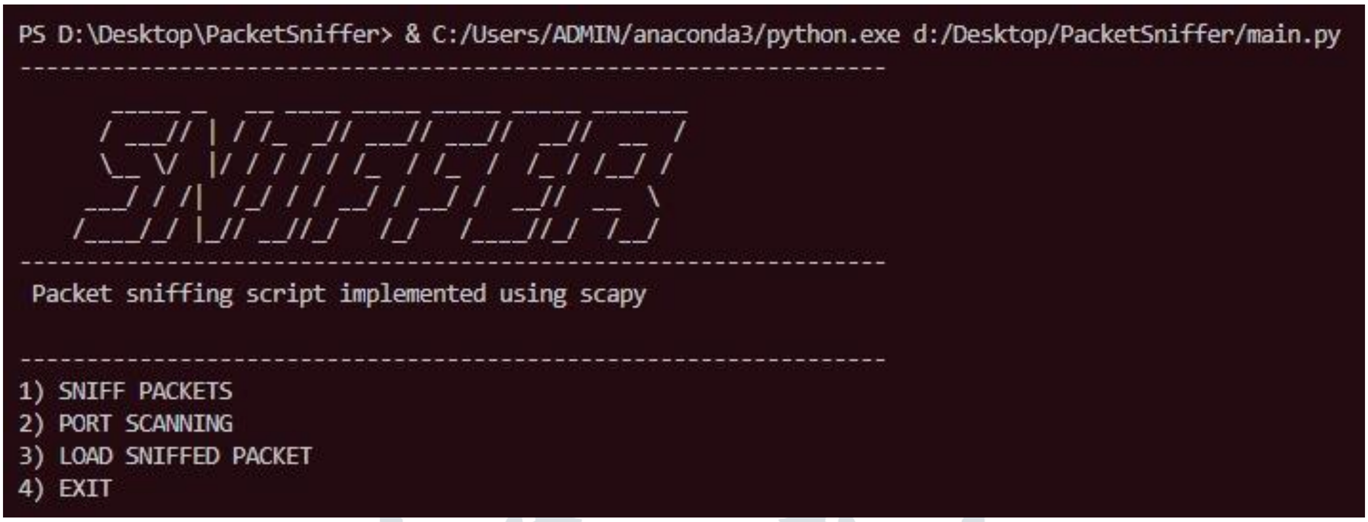


pAiNop23 Update README.md		e64d6f8 2 hours ago 8 commits
scapy	Delete scapy/scapy files	3 hours ago
LICENSE	Initial commit	4 hours ago
README.md	Update README.md	2 hours ago
main.py	Add files via upload	3 hours ago

Fig1:Files Uploaded on Github

4.2 Functioning/Working

The program does 5 things: grabbing packet, printing the content of packets with or without filters, view packet data, port scanning & load the sniffed packets



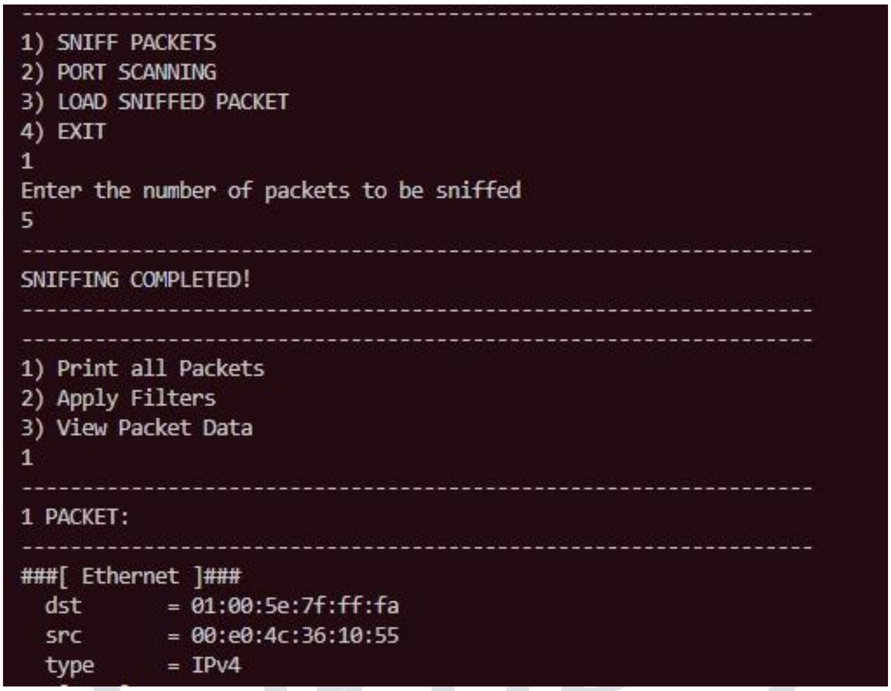
```
PS D:\Desktop\PacketSniffer> & C:/Users/ADMIN/anaconda3/python.exe d:/Desktop/PacketSniffer/main.py
--------------------------------------------------------------
      _____ _   __ ____ _____ _____ _____ _______
     / ___// | / /_  _// ___// ___// ___// __  /
     \__ \/  |/ / / / / /_  / /_  / __/ / /_/ /
    ___/ / /|  / / / / __/ / __/ / /___/ _, _/
   /____/_/ |_/ /___/_/   /_/   /_____/_/ |_|
--------------------------------------------------------------
 Packet sniffing script implemented using scapy

--------------------------------------------------------------
1) SNIFF PACKETS
2) PORT SCANNING
3) LOAD SNIFFED PACKET
4) EXIT
```

Fig 2:Menu of the Program

to grab packets we use Sniff method from scapy library which takes the following argument

- count: Number of packets to capture. 0 means infinity.
- Iface: sniff for packets only on the provided interface
- prn: Function to apply to each packet. If something is returned, it is displayed. For instance, you can use prn = lambda x: x.summary().
- store: Whether to store sniffed packets or discard them. When you only want to monitor your network forever, set store to 0.
- timeout: Stop sniffing after a given time (default: None).

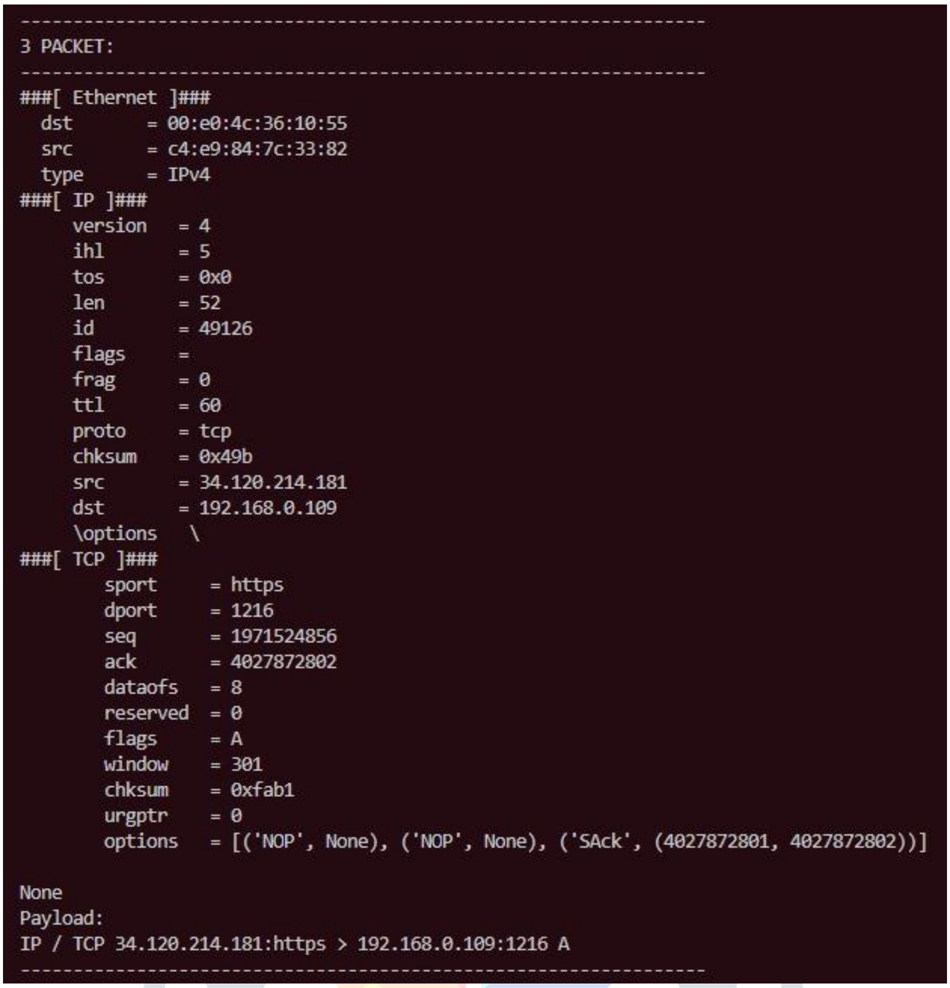


```
-----------------------------------------------------------------
1) SNIFF PACKETS
2) PORT SCANNING
3) LOAD SNIFFED PACKET
4) EXIT
1
Enter the number of packets to be sniffed
5
-----------------------------------------------------------------
SNIFFING COMPLETED!
-----------------------------------------------------------------
-----------------------------------------------------------------
1) Print all Packets
2) Apply Filters
3) View Packet Data
1
-----------------------------------------------------------------
1 PACKET:
-----------------------------------------------------------------
###[ Ethernet ]###
  dst       = 01:00:5e:7f:ff:fa
  src       = 00:e0:4c:36:10:55
  type      = IPv4
```

Fig 3:Output when option 1 is selected

For this project we have only used the ‘count’ argument

The packets are printed out using ‘show()’ method which prints out the complete information of the packet and every detail about each layer in the packet. This can be swapped with ‘summary()’ method to show relevant information only

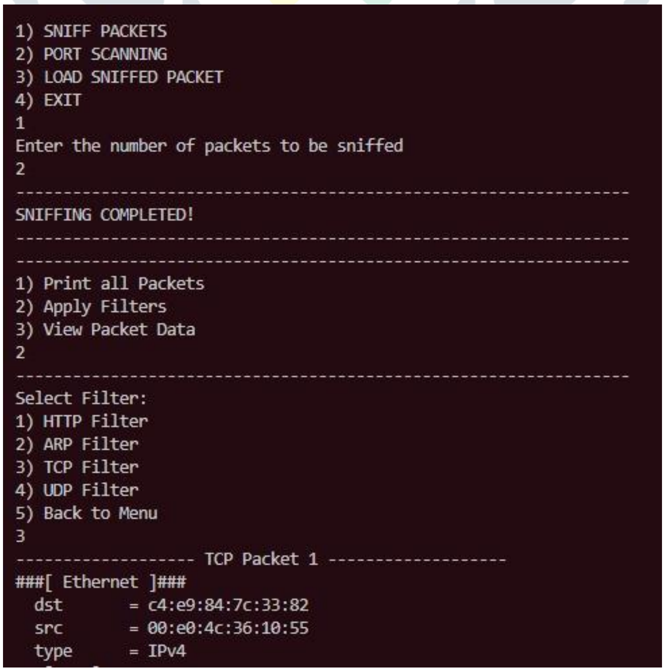


```
-----------------------------------------------------------------
3 PACKET:
-----------------------------------------------------------------
###[ Ethernet ]###
  dst       = 00:e0:4c:36:10:55
  src       = c4:e9:84:7c:33:82
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 49126
     flags     =
     frag      = 0
     ttl       = 60
     proto     = tcp
     chksum    = 0x49b
     src       = 34.120.214.181
     dst       = 192.168.0.109
     \options   \
###[ TCP ]###
        sport     = https
        dport     = 1216
        seq       = 1971524856
        ack       = 4027872802
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 301
        chksum    = 0xfab1
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('SAck', (4027872801, 4027872802))]

None
Payload:
IP / TCP 34.120.214.181:https > 192.168.0.109:1216 A
-----------------------------------------------------------------
```

Fig 4: Example output for displaying sniffed packets

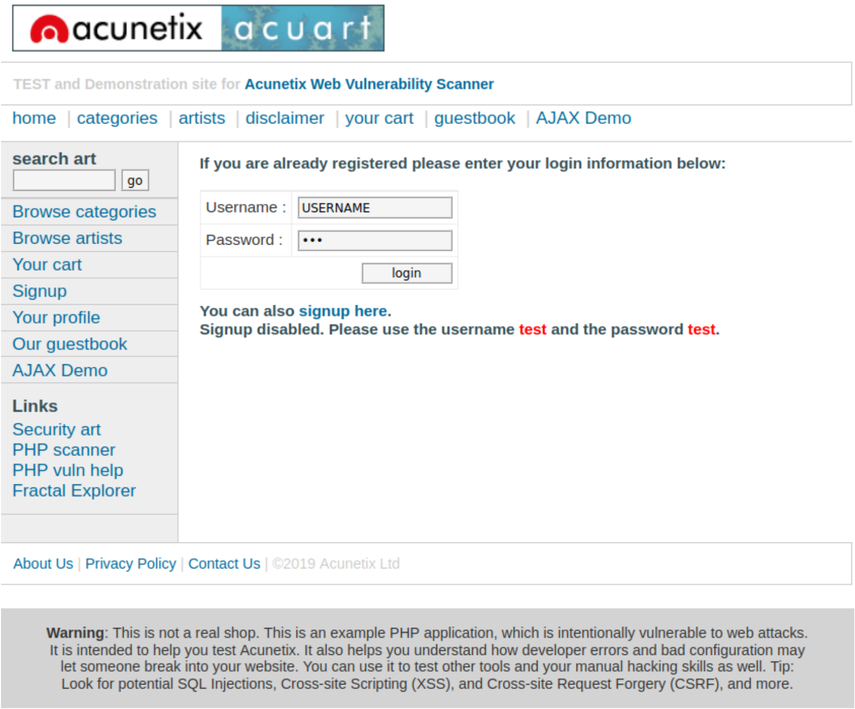
All the packets are stored in a list and then we are given the option of either displaying the all the packets or use a filter to display the contents of http packets only.



```
1) SNIFF PACKETS
2) PORT SCANNING
3) LOAD SNIFFED PACKET
4) EXIT
1
Enter the number of packets to be sniffed
2
-----------------------------------------------------------------
SNIFFING COMPLETED!
-----------------------------------------------------------------
-----------------------------------------------------------------
1) Print all Packets
2) Apply Filters
3) View Packet Data
2
-----------------------------------------------------------------
Select Filter:
1) HTTP Filter
2) ARP Filter
3) TCP Filter
4) UDP Filter
5) Back to Menu
3
------------------ TCP Packet 1 ------------------
###[ Ethernet ]###
  dst       = c4:e9:84:7c:33:82
  src       = 00:e0:4c:36:10:55
  type      = IPv4
```

Fig 5: Options to print all packets or apply filter

Now to demonstrate MiTM ,we will use a dummy http login page ,which just sends out http packets of the login credentials we put in.



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art
go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

If you are already registered please enter your login information below:

Username : USERNAME
Password : •••
login

You can also signup here.
Signup disabled. Please use the username test and the password test.

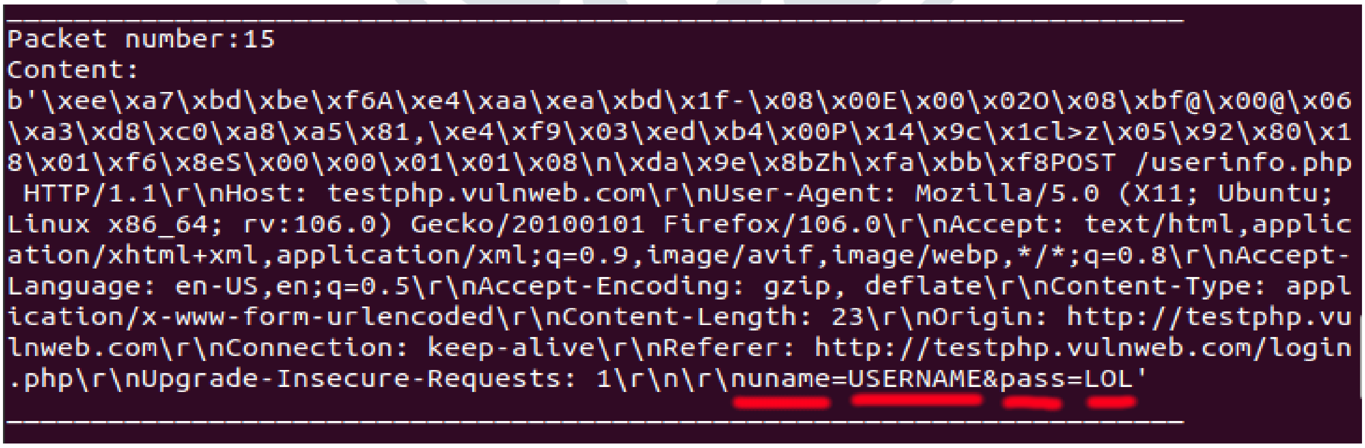
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Fig 6:Dummy HTTP login Page

Now we run the packet sniffer and enter the login button. A HTTP packet will be sent out and our program will grab that along with many other packets, we then use the filter to

only see the http packets and check the contents of every filtered packet to find the username and password.



```
______________________________________________________________________________
Packet number:15
Content:
b'\xee\xa7\xbd\xbe\xf6A\xe4\xaa\xea\xbd\x1f-\x08\x00E\x00\x02O\x08\xbf@\x00@\x06
\xa3\xd8\xc0\xa8\xa5\x81,\xe4\xf9\x03\xed\xb4\x00P\x14\x9c\x1cl>z\x05\x92\x80\x1
8\x01\xf6\x8eS\x00\x00\x01\x01\x08\n\xda\x9e\x8bZh\xfa\xbb\xf8POST /userinfo.php
 HTTP/1.1\r\nHost: testphp.vulnweb.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu;
Linux x86_64; rv:106.0) Gecko/20100101 Firefox/106.0\r\nAccept: text/html,applic
ation/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-
Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nContent-Type: appl
ication/x-www-form-urlencoded\r\nContent-Length: 23\r\nOrigin: http://testphp.vu
lnweb.com\r\nConnection: keep-alive\r\nReferer: http://testphp.vulnweb.com/login
.php\r\nUpgrade-Insecure-Requests: 1\r\n\r\nuname=USERNAME&pass=LOL'
______________________________________________________________________________
```

Fig 7:Output when HTTP filter is applied on the credential containing packet

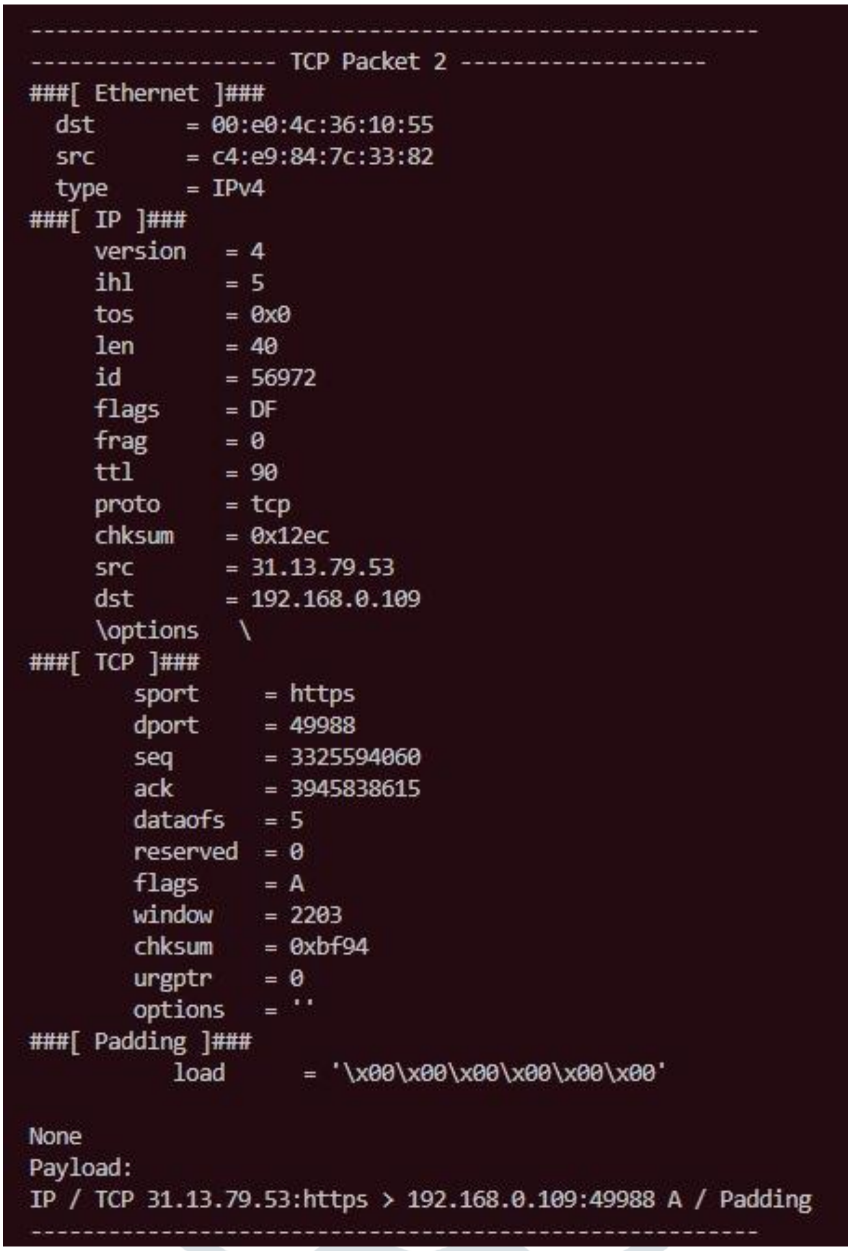
the 15th packet that we sniffed out had the credentials, in plain unencrypted text. This was the MiTM attack

Now on to Port scanning. This program checks on port number 20, 21, 22, 143, 443, 5060

1. 20-22 (TCP): A communication standard for delivering data and messages through networks
2. 20-22 (UDP) : A core communication protocol that sends messages to other hosts on an IP
3. 20-21 (FTP): A client server protocol that allows a client to request a file and the server to supply it
4. 22 (SSH): A network protocol that provides a secure way to access a computer over an unsecured network
5. 143 (IMAP): A protocol that allows the user to view and edit messages without downloading them
6. 5060 (SIP): A signaling protocol that controls multimedia communication sessions over the internet
7. 443-Hypertext transfer Protocol Secured (HTTPS)

More port numbers can be added in the program by editing the port number list

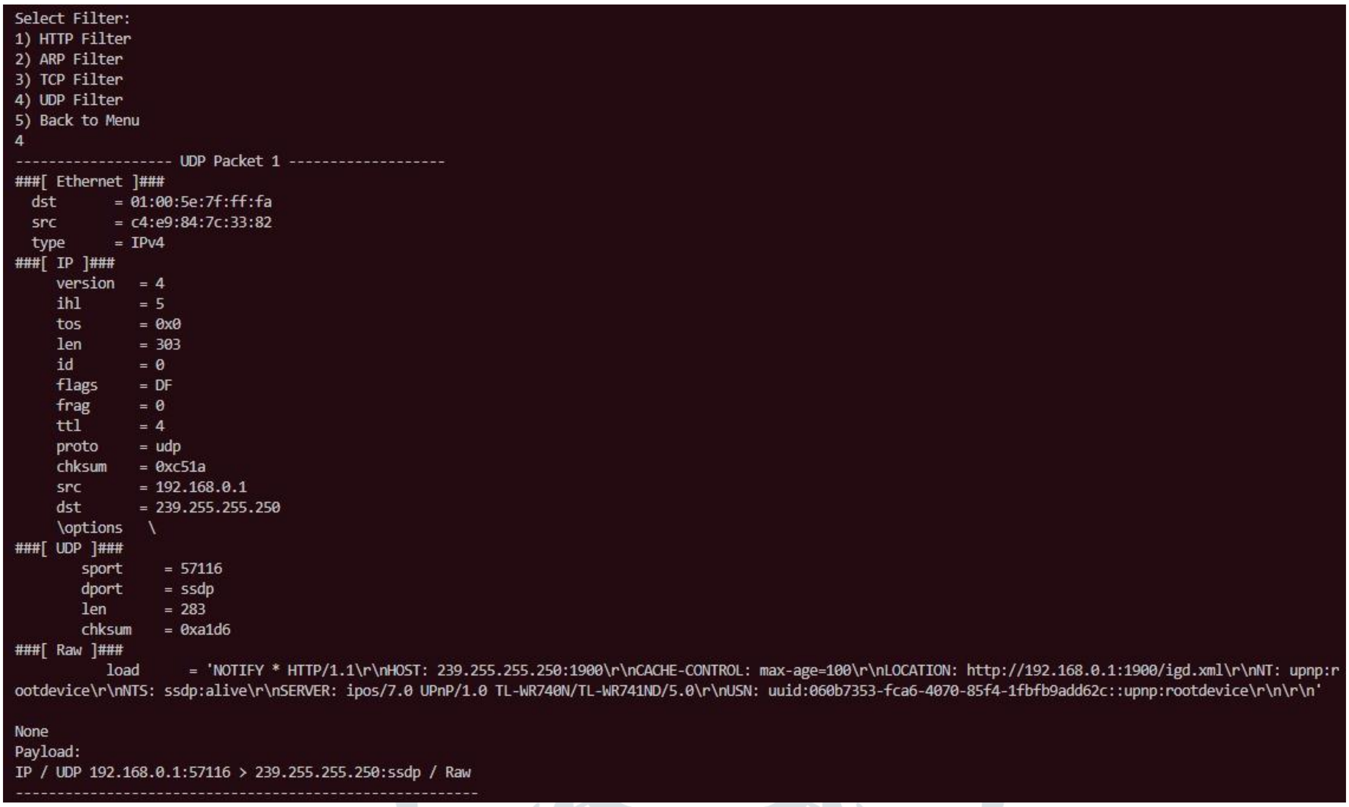
The program first makes a custom packet using Scapy, and then sends it to the destination via 'SR1()' method, which sends a packet to the source and waits for the reply. Then an If else block is added which checks the reply, If the reply is none then the port is filtered.



```
--------------------------------------------------------
------------------- TCP Packet 2 -------------------
###[ Ethernet ]###
  dst       = 00:e0:4c:36:10:55
  src       = c4:e9:84:7c:33:82
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 56972
     flags     = DF
     frag      = 0
     ttl       = 90
     proto     = tcp
     chksum    = 0x12ec
     src       = 31.13.79.53
     dst       = 192.168.0.109
     \options   \
###[ TCP ]###
        sport     = https
        dport     = 49988
        seq       = 3325594060
        ack       = 3945838615
        dataofs   = 5
        reserved  = 0
        flags     = A
        window    = 2203
        chksum    = 0xbf94
        urgptr    = 0
        options   = ''
###[ Padding ]###
           load      = '\x00\x00\x00\x00\x00\x00'

None
Payload:
IP / TCP 31.13.79.53:https > 192.168.0.109:49988 A / Padding
--------------------------------------------------------
```

Fig 8: Output when option 3 is selected

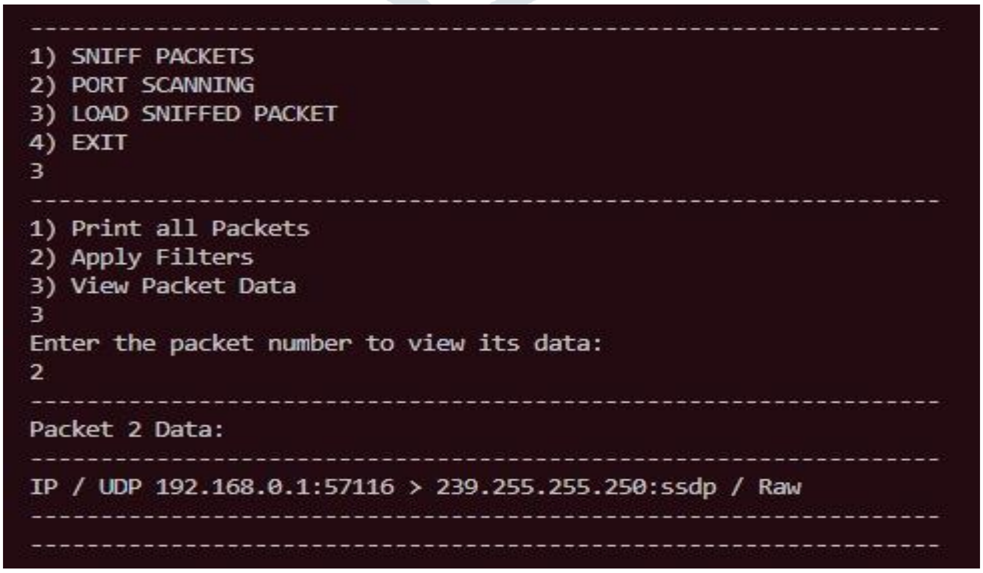


```
Select Filter:
1) HTTP Filter
2) ARP Filter
3) TCP Filter
4) UDP Filter
5) Back to Menu
4
------------------ UDP Packet 1 ------------------
###[ Ethernet ]###
  dst       = 01:00:5e:7f:ff:fa
  src       = c4:e9:84:7c:33:82
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 303
     id        = 0
     flags     = DF
     frag      = 0
     ttl       = 4
     proto     = udp
     chksum    = 0xc51a
     src       = 192.168.0.1
     dst       = 239.255.255.250
     \options   \
###[ UDP ]###
        sport     = 57116
        dport     = ssdp
        len       = 283
        chksum    = 0xa1d6
###[ Raw ]###
           load      = 'NOTIFY * HTTP/1.1\r\nHOST: 239.255.255.250:1900\r\nCACHE-CONTROL: max-age=100\r\nLOCATION: http://192.168.0.1:1900/igd.xml\r\nNT: upnp:r
ootdevice\r\nNTS: ssdp:alive\r\nSERVER: ipos/7.0 UPnP/1.0 TL-WR740N/TL-WR741ND/5.0\r\nUSN: uuid:060b7353-fca6-4070-85f4-1fbfb9add62c::upnp:rootdevice\r\n\r\n'

None
Payload:
IP / UDP 192.168.0.1:57116 > 239.255.255.250:ssdp / Raw
--------------------------------------------------------
```

Fig 9: Output when option filter is applied

If a packet is received then we check its ‘Flag’ attribute in the TCP layer, if it is ‘0x12’ then the port is open otherwise its closed and if the packet has an ICMP layer then we check if its ‘type’ attribute is 3 and ‘code’ attribute is 1,2,3,9,10 or 13; if these conditions are true then the port is filtered.



```
----------------------------------------------------------------
1) SNIFF PACKETS
2) PORT SCANNING
3) LOAD SNIFFED PACKET
4) EXIT
3
----------------------------------------------------------------
1) Print all Packets
2) Apply Filters
3) View Packet Data
3
Enter the packet number to view its data:
2
----------------------------------------------------------------
Packet 2 Data:
----------------------------------------------------------------
IP / UDP 192.168.0.1:57116 > 239.255.255.250:ssdp / Raw
----------------------------------------------------------------
----------------------------------------------------------------
```

Fig 10: Output when option to View Packet Data is applied

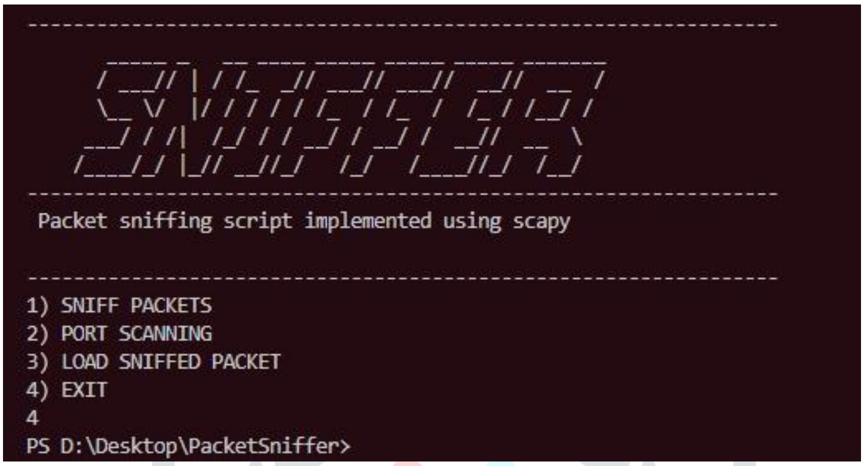
A loop iterates through each packet, displaying the packet number and using k.show() to show the packet details.

Each packet's payload is printed using k.payload after the packet data are displayed.

The user is asked to provide the packet number if they want to access specific packet data.

Next, selected_packet.payload is used to display the payload of the selected packet.

And lastly the exit option allows you to quit the program



```
---------------------------------------------------------------

     _____ _   __ ____ ______ ______ ______ ______
    / ___// | / //  _// ____// ____// ____// __  /
    \__ \/  |/ / / / / /_   / /_   / __/  / /_/ /
   ___/ / /|  /_/ / / __/  / __/  / /___ / _, _/
  /____/_/ |_//___//_/    /_/    /_____//_/ |_|
---------------------------------------------------------------
 Packet sniffing script implemented using scapy

---------------------------------------------------------------
1) SNIFF PACKETS
2) PORT SCANNING
3) LOAD SNIFFED PACKET
4) EXIT
4
PS D:\Desktop\PacketSniffer>
```

Fig 11: Output when exited

Chapter 5

Conclusion

5.1 Conclusion

In conclusion, all the group members have learnt Java and Python, along with some basic concepts of Networks.

Our Literature Team Devina Panchal, Yogeshchand Rai, Swaraj Sakpal & Vansh Visariya led by our Group Leader Devina Panchal are responsible for the picking, developing and completion of this Project & Documentation. From Gathering Information to working on the same, we hereby believe to have acquired some extent of Knowledge of Scapy Library & its Applications.

5.2 Future Extent

Simplified User Interface:

1) Refine the user interface for a more intuitive experience.
2) Enhance menu options and overall navigation.

Advanced Packet Analysis:

1) Implement deeper packet inspection for more detailed insights.
2) Add functionality for analyzing specific protocols or patterns. ast but not least, to increase functionality in response to client feedback.

Community Collaboration:

1) Open-source the project on GitHub for community contributions.
2) Encourage feedback and collaboration for continuous improvement.

5.3 Limitations

1) The current Program cannot be executed on your device without remote scapy and Python installation.
2) Only two essential functionalities are supported in the absence of a comprehensive GUI.

Chapter 6

Reference

[1] E. V. Ananin, A. V. Nikishova and I. S. Kozhevnikova, "Port scanning detection based on anomalies," 2017 Dynamics of Systems, Mechanisms and Machines (Dynamics), 2017, pp. 1-5, doi: 10.1109/Dynamics.2017.8239427. https://ieeexplore.ieee.org/abstract/document/8239427/

[2] M. Dabbagh, A. J. Ghandour, K. Fawaz, W. E. Hajj and H. Hajj, "Slow port scanning detection," 2011 7th International Conference on Information Assurance and Security (IAS), 2011, pp. 228-233, doi: 10.1109/ISIAS.2011.6122824. https://ieeexplore.ieee.org/abstract/document/6122824/

[3] M. Shah, S. Ahmed, K. Saeed, M. Junaid, H. Khan and Ata-ur-rehman, "Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool," 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019, pp. 1-6, doi: 10.1109/ICOMET.2019.8673520. https://ieeexplore.ieee.org/abstract/document/8673520/

[4] W. El-Hajj, F. Aloul, Z. Trabelsi and N. Zaki, "On Detecting Port Scanning using Fuzzy Based Intrusion Detection System," 2008 International Wireless Communications and Mobile Computing Conference, 2008, pp. 105-110, doi: 10.1109/IWCMC.2008.19. https://ieeexplore.ieee.org/abstract/document/4599918/

[5] F. H. Mohd Ali, R. Yunos and M. A. Mohamad Alias, "Simple port knocking method: Against TCP replay attack and port scanning," Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012, pp. 247-252, doi: 10.1109/CyberSec.2012.6246118. https://ieeexplore.ieee.org/abstract/document/6246118/

[6] A. K. Kaushik, E. S. Pilli and R. C. Joshi, "Network forensic system for port scanning attack," 2010 IEEE 2nd International Advance Computing Conference (IACC), 2010, pp. 310-315, doi: 10.1109/IADCC.2010.5422935. https://ieeexplore.ieee.org/abstract/document/5422935/

[7] F. Tang, Y. Kawamoto, N. Kato, K. Yano and Y. Suzuki, "Probe Delay Based Adaptive Port Scanning for IoT Devices with Private IP Address Behind NAT," in IEEE Network, vol. 34, no. 2, pp. 195-201, March/April 2020, doi: 10.1109/MNET.001.1900264. https://ieeexplore.ieee.org/abstract/document/8869708/

[8] G. Nath Nayak and S. Ghosh Samaddar, &quot;Different flavours of Man-In-The-Middle attack, consequences and feasible solutions,&quot; 2010 3rd International Conference on Computer Science and Information Technology, 2010, pp. 491-495, doi: 10.1109/ICCSIT.2010.5563900. https://ieeexplore.ieee.org/abstract/document/5563900

[9] J. Huang, Y. Wang, H. Wang, Z. Li and J. Huang, &quot;Man-in-the-middle attack on BB84 protocol and its defence,&quot; 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 438-439, doi: 10.1109/ICCSIT.2009.5234678. https://ieeexplore.ieee.org/abstract/document/5234678

[10] O. Salem, K. Alsubhi, A. Shaafi, M. Gheryani, A. Mehaoua and R. Boutaba, &quot;Man-in-the-Middle Attack Mitigation in Internet of Medical Things,&quot; in IEEE Transactions on Industrial Informatics, vol. 18, no. 3, pp. 2053-2062, March 2022, doi: 10.1109/TII.2021.3089462. https://ieeexplore.ieee.org/abstract/document/9456085

[11] M. Hayashi and Á. Vázquez-Castro, &quot;Physical Layer Security Protocol for Poisson Channels for Passive Man-in-the-Middle Attack,&quot; in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2295-2305, 2020, doi: 10.1109/TIFS.2019.2963771. https://ieeexplore.ieee.org/abstract/document/8949690

[12] L. Zhang, W. Jia, S. Wen and D. Yao, &quot;A Man-in-the-Middle Attack on 3G-WLAN Interworking,&quot; 2010 International Conference on Communications and Mobile Computing, 2010, pp. 121-125, doi: 10.1109/CMC.2010.34. https://ieeexplore.ieee.org/abstract/document/5471501

[13] J. Du, X. Li and H. Huang, &quot;A Study of Man-in-the-Middle Attack Based on SSL Certificate Interaction,&quot; 2011 First International Conference on

Instrumentation, Measurement, Computer, Communication and Control, 2011, pp. 445-448, doi: 10.1109/IMCCC.2011.117.
https://ieeexplore.ieee.org/abstract/document/6154142

[14] Z. Chen, S. Guo, K. Zheng and Y. Yang, &quot;Modeling of Man-in-the-Middle Attack in the Wireless Networks,&quot; 2007 International Conference on Wireless Communications, Networking and Mobile Computing, 2007, pp. 2255-2258, doi: 10.1109/WICOM.2007.562.1
https://ieeexplore.ieee.org/abstract/document/434037

[15] Z. Trabelsi, H. Rahmani, K. Kaouech and M. Frikha, &quot;Malicious sniffing systems detection platform,&quot; 2004 International Symposium on Applications and the Internet. Proceedings., 2004, pp. 201-207, doi: 10.1109/SAINT.2004.1266117.
https://ieeexplore.ieee.org/abstract/document/1266117

[16] J. Broadway, B. Turnbull and J. Slay, &quot;Improving the Analysis of Lawfully Intercepted Network Packet Data Captured for Forensic Analysis,&quot; 2008 Third International Conference on Availability, Reliability and Security, 2008, pp. 1361-1368, doi: 10.1109/ARES.2008.122.
https://ieeexplore.ieee.org/abstract/document/4529503

[17] A.Dabir and A. Matrawy, &quot;Bottleneck Analysis of Traffic Monitoring using Wireshark,&quot; 2007 Innovations in Information Technologies (IIT), 2007, pp. 158-162, doi: 10.1109/IIT.2007.4430446.
https://ieeexplore.ieee.org/abstract/document/4430446

[18] G. Jinhua and X. Kejian, &quot;ARP spoofing detection algorithm using ICMP protocol,&quot; 2013 International Conference on Computer Communication and Informatics, 2013, pp. 1-6, doi: 10.1109/ICCCI.2013.64662900.
https://ieeexplore.ieee.org/abstract/document/6466290

[19] J. Janitor, F. Jakab and K. Kniewald, &quot;Visual Learning Tools for Teaching/Learning Computer Networks: Cisco Networking Academy and Packet Tracer,&quot; 2010 Sixth International Conference on Networking and Services, 2010, pp. 351-355, doi: 10.1109/ICNS.2010.55.
https://ieeexplore.ieee.org/abstract/document/5460623

[20] P. Anu and S. Vimala, &quot;A survey on sniffing attacks on computer networks,&quot; 2017 International Conference on Intelligent Computing and Control (I2C2), 2017, pp. 1-5, doi: 10.1109/I2C2.2017.8321914
https://ieeexplore.ieee.org/abstract/document/8321914
[21] S. Ansari, S. G. Rajeev and H. S. Chandrashekar, "Packet sniffing: a brief introduction," in IEEE Potentials, vol. 21, no. 5, pp. 17-19, Dec. 2002-Jan. 2003, doi: 10.1109/MP.2002.1166620.
https://ieeexplore.ieee.org/abstract/document/1166620/

[22] M. A. Qadeer, A. Iqbal, M. Zahid and M. R. Siddiqui, "Network Traffic Analysis and Intrusion Detection Using Packet Sniffer," 2010 Second International Conference on Communication Software and Networks, 2010, pp. 313-317, doi: 10.1109/ICCSN.2010.104.
https://ieeexplore.ieee.org/abstract/document/5437681/

[23] 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.
https://ieeexplore.ieee.org/abstract/document/8319360/

[24] A. Siswanto, A. Syukur, E. A. Kadir and Suratin, "Network Traffic Monitoring and Analysis Using Packet Sniffer," 2019 International Conference on Advanced Communication Technologies and Networking (CommNet), 2019, pp. 1-4, doi: 10.1109/COMMNET.2019.8742369.
https://ieeexplore.ieee.org/abstract/document/8742369/

Acknowledgement

We take this opportunity to express our sincere thanks to our Guide, Ms. PRANALI PAWAR & Co-Guide Ms.ASHA DURAFE the faculty in the Department of Cyber Security in Shah and Anchor Kutchhi Engineering College for guiding us and suggesting regarding the line of work for our project “MiTM Using Custom Packet Sniffer”. We would like to express our gratitude towards their constant encouragement, support and guidance throughout the progress.

Also, we would like to thank our Principal – Dr. Bhavesh Patel and Dr. Nilakshi Jain, Head of Cyber Security Department, for their help, support & guidance for this project.

We are also thankful to all Faculty members of our department for their help and guidance during completion of our project.