Original Research Paper

Enhanced Techniques for Detecting Promiscuous Mode using Packet Fu and the Metasploit Framework

Partho Pandya[1], Kashyap Joshi[1*], Kapil Kumar[1]

[1] *Department of Cybersecurity and Forensic, Department of Biochemistry and Forensic Science, Gujarat University. Ahmedabad, India.*

Article History
Received:
17.07.2024

Revised:
30.07.2025

Accepted:
11.08.2025

***Corresponding Author:**
Kashyap Joshi
Email:
kashyapjoshi.it@gmail.com

This is an open access article, licensed under: CC–BY-SA

Abstract: The detection of promiscuous mode is an essential component of modern network security since it allows for the prevention of possible attacks. Packet Fu is a powerful network analysis package that is being utilised in this study to investigate the identification of promiscuous mode operation. Within the scope of this study, the idea of promiscuous mode, its consequences, and the significance of its detection in the context of preserving network security are all investigated. This article provides an in-depth analysis of a variety of detection techniques, showing both their merits and limitations. In addition to this, it presents a unique method for robust detection that makes use of Packet Fu and is validated by empirical data. Through the provision of fresh insights into detection approaches and the provision of creative solutions for efficient threat interception, the purpose of this study is to improve the security of the network.

Keywords: Detection Methodologies, Network Security, Packet Fu, Promiscuous Mode, Threat Interception.

1. Introduction

The Promiscuous mode is a crucial feature of network interface controllers (NICs) that allows a device to monitor and analyze all network traffic, not just the traffic intended for it. This capability is essential for network diagnostics and legitimate monitoring activities. However, it also presents significant security risks as it can be exploited for unauthorized network surveillance and data interception, thereby facilitating malicious activities.

Detecting the use of promiscuous mode is vital for identifying potential security breaches and fortifying network defenses. By identifying devices operating in promiscuous mode, network administrators can pinpoint unauthorized monitoring attempts and take necessary actions to mitigate the risk. This detection process involves analyzing network traffic patterns and utilizing specialized techniques to uncover any device that is intercepting and processing packets it should not normally receive. Enhancing network security through the detection of promiscuous mode thus becomes a fundamental step in safeguarding network assets and ensuring the integrity and confidentiality of data transmissions.

2. Literature Review

Promiscuous mode detection has garnered significant attention in network security research. Various techniques and tools have been developed for detection. However, existing methods may be susceptible to evasion tactics, necessitating exploration of innovative methodologies, The Research addresses the Destructive problem of Denial-of-Service (DoS) assaults, which represent substantial threats to network infrastructures, particularly as our reliance on computers and communication networks increases. It examines the many forms of Denial-of-Service (DoS) and Distributed Denial-of Service (DDoS) attacks, emphasizing their disruptive character and the possible harm they can do to network security and availability. Because DoS/DDoS attacks are dynamic, traditional methods that use machine learning for identifying them are shown to be inadequate. Therefore, using the algorithms of the Support Vector Machine (SVM), Multi-Layer Perceptron (the MLP), and long-short-term memory (LSTM), the study proposes a Deep Learning model. When it comes to discerning typical network traffic and patterns of attack, our model performs remarkably well. In order to lessen attacks, it also uses filtering of traffic and rate-limiting mechanisms. Using the Library of SCAPY python module, with the help of this you will be able to do it in the required python environment options [1].

An approach to cybersecurity tailored for wireless networks in smart electrical substations will be discussed in this article. This technique uses WPA3, which provides encryption, authentication, and integrity, for layer 2 security. A Cooperative Hybrid Intrusion Detection System (CHIDS) is used to find possible threats by using four different detection methods: parameter evaluation, anomalies, signatures, and access control. Additionally, the plan calls for securely distributing keys via a local authenticating server rather than a remote RADIUS server. Simulation results show that these security precautions only slightly increase latency, remaining below the 4 milliseconds barrier needed for real-time protection applications. The model's overall goal is to meet the demands of performance while guaranteeing the safe functioning of wireless networks for communication inside substations throughout the Work Flow Goes [2].

In order to locate network sniffers inside a certain network infrastructure, this article suggests using mobile agents. Specialized programs, also known as mobile agents, are programmed that may relocate and operate at different nodes in a network. They act as messengers sent by a network administrator to collect relevant information from many computer systems scattered over the network structure. By examining this gathered data, network managers can identify whether any computers are running in promiscuous mode, which could mean that a sniffer program [14] is secretly monitoring network traffic. Their deft navigation of the intricate network infrastructure's maze-like passageways embodies the essence of a clandestine operative. Strategically deployed by the watchful network steward, these special agents go on reconnaissance missions, surreptitiously monitoring and scrutinizing the operations of every system they come across. Their operational mandate involves tracking system performance closely and focusing on any irregularities that would be signs of possible sniffer activity. Promiscuous mode activation detection is critical because it allows a system to covertly collect and examine all incoming and outgoing data packets, regardless of who the intended recipients are. After completing their surreptitious reconnaissance, these mobile agents return with a wealth of intelligence to the network administrator. As the chief defender of network security, the administrator is aware & eagerly expecting their report. The administrator discovers patterns in some of the purportedly compromised systems that strongly indicate they have been operating in a promiscuous manner after reviewing the data that has been gathered. This is a blatant sign that ought to compel the administrator to look into it more. Equipped with such helpful data, the administrator proceeds with an immediate examination, ensuring that any tools he finds that are unique to the sniffer are genuine. In conclusion, mobile agents have shown to be valuable companions in the ongoing fight to defend the network architectures against unseen intrusions, acting as a deterrent fueled by the stealthy techniques employed by network sniffers. RTBU. The network owner can stay in charge of the network thanks to their skill in changing the host environment's network variables and surreptitiously manufacturing potions [5].

When accessed and presented, these embedded contents may have unexpected side repercussions like the execution of malicious JavaScript code that could reveal sensitive information. In this scenario, if the uploading of malicious files could be prevented from the server end right away, the primary driver of these attacks could be turned off. Nevertheless, the current server side's use of content sniffing for attack detection has a number of drawbacks. An attacker can save the pay load data wherever in a file since, first of all, the content of a file is examined only up to a predetermined number of beginning bytes. Secondly, these methods lack a mechanism to assess the level of harm that malicious content does to different browsers. In order to solve these problems, this work creates a content parsing-based server-side content attacker detection system [4].

Packet sniffing is a technology used to monitor network traffic, as discussed in the paper "Detecting Packets Using Packet Sniffing." Network traffic analysis devices are becoming increasingly essential as computer networks get bigger and more intricate. The authors give an explanation of how packet sniffers operate in switched and non-switched structures and provide many examples of applications for the hardware and software. They also go over the benefits and drawbacks of various sniffing methods. They go on to talk about packet sniffing in more detail, including how one can employ it to record and examine network data. Sniffing techniques such as ARP cache poisoning, CAM Table flooding, and switch port hijacking are presented. The paper also addresses several kinds of techniques for MAC-based, ARP-Based, NETWORK BASED sniffing [6].

Phan and Park created an adaptable SDN-based architecture that employs machine learning to identify and stop low-rate DDoS attacks. By combining a hybrid machine learning model with the enhanced HIPF (eHIPF) scheme, their method presents a novel protection mechanism for SDN-based cloud environments, substantially improving the ability of the framework to recognize and address such threats [22].

Because of their dispersed nature and high processing demands, detecting dispersed Denial of Service, or DDoS, assaults in cloud settings is a difficult task. In contrast to a DoS attack [18], a distributed denial-of-service (DDoS comes from several sources and utilizes system and network resources to prevent legitimate users from reaching the system. These assaults often take place at the network and application levels, among other OSI layers. The goal of recent research is to identify malicious traffic patterns by treating DDoS detection as a machine learning classification issue. Multiple regression models have been investigated to forecast DDoS and bot assaults, especially during periods of high traffic, using the CICIDS 2017 dataset [23].

To improve the precision and stability of DDoS attack detection, academics have suggested a hybrid ensemble learning approach which utilizes the use of Singular Value Decomposition (SVD). This method shows better performance and dependability than traditional approaches like Random Forest and k-NN [24]

The study dealt with improving, training, and testing the performance of a Distributed Time Delay Neural Network (DTDNN) for intrusion detection. DTDNN successfully distinguished between different attack types, including DoS, U2R, R2L, and Probe, with a high identification rate of 97.24%, according to experiments performed on the KDD99 dataset. Compared to conventional static network models, its memory-based structure and dynamic ability to learn allowed for faster convergence and higher accuracy [25].

Network security is still highly endangered by Distributed Denial of Service, or DDoS, attacks, and researchers utilize machine learning in increasing numbers to discover and stop them. Perez-Diaz et al. investigated a number of intelligent detection techniques, focusing on anomaly categorization and traffic pattern identification. According to their research, incorporating a passive sniffer technique—which keep an eye on and evaluates network packets without changing traffic flow—can improve early warning accuracy while reducing system chaos, making it an advantageous approach for real-time DDoS mitigation [26].

The network is not overloaded with traffic after an LR-DDoS attack. Rather, it deliberately begins key protocol operations such TCP's timeout retransmission [27].

Cross-site scripting (XSS) represents one of the biggest risks to web applications, largely because of inadequate input sanitization and the growing use of dynamic user-generated content. By considering XSS vulnerabilities, SWAP (Secure Web Application Proxy), a server-side solution that uses a reverse proxy to gather and examine HTML data in order to successfully identify real XSS vulnerabilities in well-known web apps while in internal network it is more dangerous, the attack can escalate to MITM takeovers and Credential Stuffing [28].

Everyone can comprehend the method that is used to determine packets of data in a non-switched environment. Every host in this technology is linked to a hub. Numerous commercial and non-commercial solutions are available that enable listening in on network communications. The issue of how this network communication can be eavesdropped off now arises; this issue can be resolved by putting the network card in an individual "promiscuous mode" [29].

Because network sniffers can be used fraudulently, data transmission security between source and destination is crucial in layered network topologies. Sniffers are frequently used for monitoring, troubleshooting, and analysis, but if they are not adequately regulated, they can also result in protocol abuse. Network security is greatly improved by putting safe and strong SSH authentication, HTTPS, and VPN tunneling into practice. This study stresses the need for more research on decrypting encrypted communication, especially in IPv6 situations and presents plausible packet sniffer strategies [30].

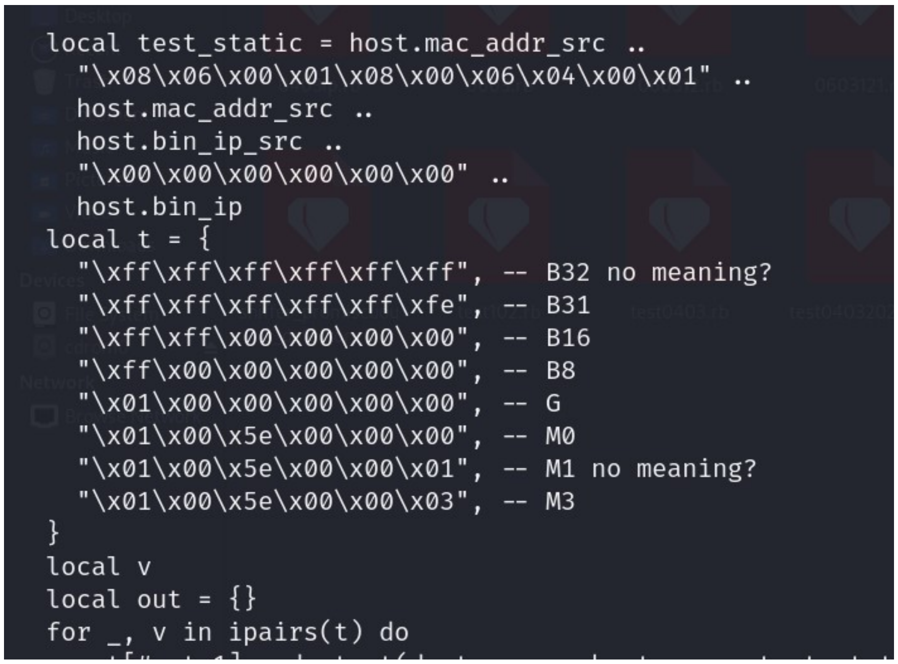
3. Methodology

The proposed methodology utilizes multiple built-in libraries of the Ruby language, specifically leveraging Packet Fu to generate specified packets. These generated packets are transmitted across the network, whereupon the responses from networked devices are analyzed to create alerts based on the received responses.

The methodology involves crafting a specific packet designed to query whether a device is operating in promiscuous mode. The response from the device will include its MAC address, indicating its status. If the crafted packet receives a response confirming the presence of a device in promiscuous mode along with the specified MAC address, an alert is generated.

This method is particularly effective because many sniffing tools, whether passive sniffing tools like Wireshark or 'tcpdump', operate in promiscuous or monitor mode. By changing the state of the NIC, these tools enable the detection mechanism to query the card's state. This approach provides a higher accuracy level compared to traditional Intrusion Detection Systems (IDS), which often fail to detect passive sniffing.

Figure 1 is illustrating the network packet created by the Packet-FU



```
local test_static = host.mac_addr_src ..
  "\x08\x06\x00\x01\x08\x00\x06\x04\x00\x01" ..
  host.mac_addr_src ..
  host.bin_ip_src ..
  "\x00\x00\x00\x00\x00\x00" ..
  host.bin_ip
local t = {
  "\xff\xff\xff\xff\xff\xff", -- B32 no meaning?
  "\xff\xff\xff\xff\xff\xfe", -- B31
  "\xff\xff\x00\x00\x00\x00", -- B16
  "\xff\x00\x00\x00\x00\x00", -- B8
  "\x01\x00\x00\x00\x00\x00", -- G
  "\x01\x00\x5e\x00\x00\x00", -- M0
  "\x01\x00\x5e\x00\x00\x01", -- M1 no meaning?
  "\x01\x00\x5e\x00\x00\x03", -- M3
}
local v
local out = {}
for _, v in ipairs(t) do
```

Figure 1. The Network Packet Created by the Packet-FU

This research proposes an innovative methodology for detection, leveraging Packet Fu. This methodology encompasses deep packet inspection, behavioral profiling, real-time signature generation, and ensemble learning.

Deep Packet Inspection, Utilizing the comprehensive feature set of Packet Fu, this methodology conducts in-depth packet inspection to identify indicators of promiscuous mode activity. It is also capable of detecting anomalous network behavior and determining whether returned packets suggest promiscuous mode. Furthermore, the response from these packets can provide detailed information about the device, enhancing the precision of detection.

The method supports multiple operating systems, making it an open-source and versatile solution. Additionally, the accuracy of this detection technique surpasses that of many IDS solutions, which do not typically detect passive sniffing. The integration with Metasploit enhances its detection capabilities and provides an alerting mechanism, ensuring precise and timely identification of devices in promiscuous mode.

4. Finding and Discussion

4.1. Finding

The steps being followed to get all the workflow and found the results after the tool deployment, are:

1) First, create the code and save it as a module in the Metasploit framework with the name *Ipbaseddetection.rb Path: /usr/share/metasploit-framework/modules/auxiliary/sniffer/*

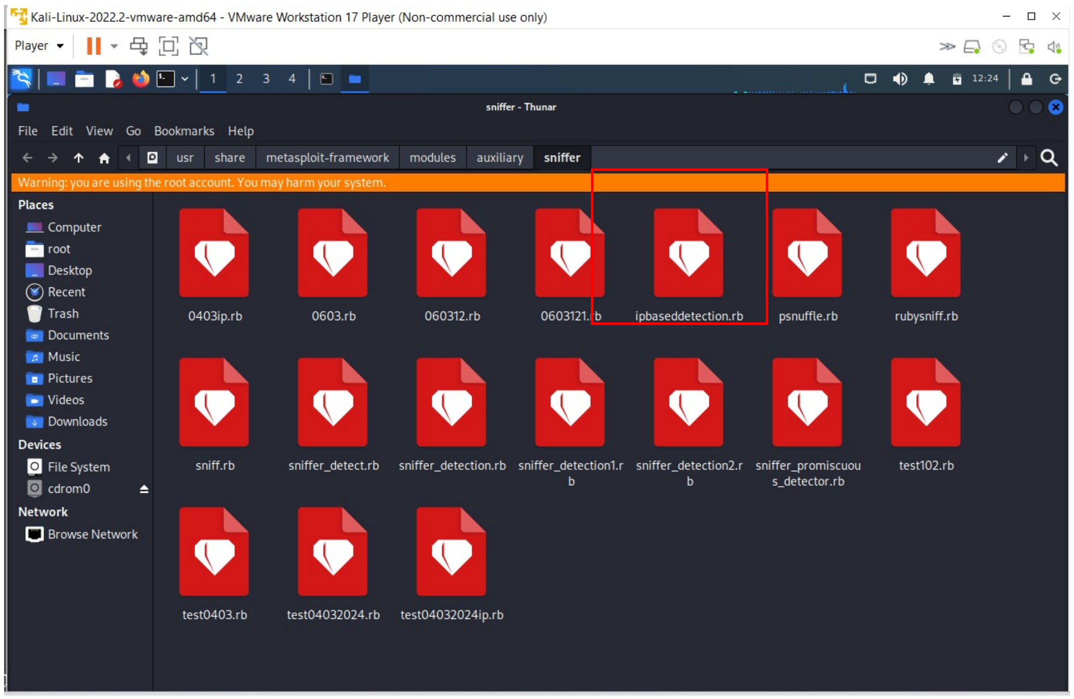
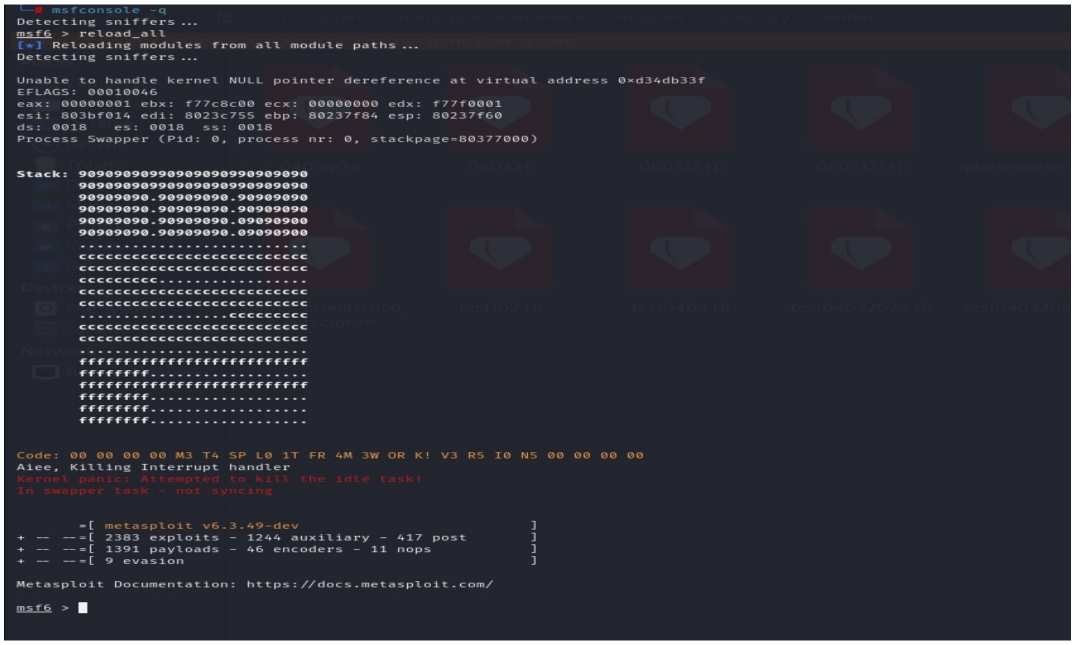


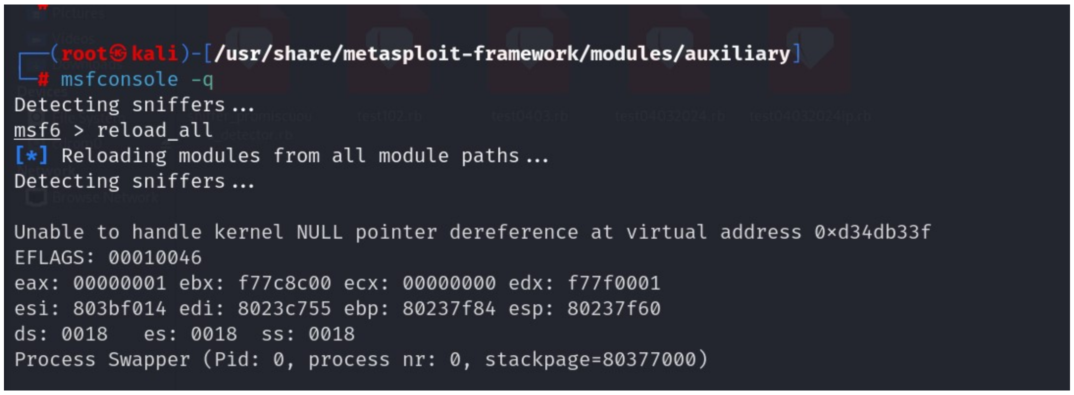
Figure 2. Ruby Script Location

2) Subsequently, reload the Metasploit framework to incorporate the newly created code into the framework. This reloading process updates the repository, ensuring that the new module is available in the Metasploit framework console.



```
└─# msfconsole -q
Detecting sniffers ...
msf6 > reload_all
[*] Reloading modules from all module paths ...
Detecting sniffers ...

Unable to handle kernel NULL pointer dereference at virtual address 0×d34db33f
EFLAGS: 00010046
eax: 00000001 ebx: f77c8c00 ecx: 00000000 edx: f77f0001
esi: 803bf014 edi: 8023c755 ebp: 80237f84 esp: 80237f60
ds: 0018   es: 0018  ss: 0018
Process Swapper (Pid: 0, process nr: 0, stackpage=80377000)


Stack: 90909090990909090990909090
       90909090990909090990909090
       90909090.90909090.90909090
       90909090.90909090.90909090
       90909090.90909090.09090900
       90909090.90909090.09090900
       ..........................
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       ccccccccc.................
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       ..................cccccccc
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       ..........................
       ffffffffffffffffffffffffff
       ffffffff..................
       ffffffffffffffffffffffffff
       ffffffff..................
       ffffffff..................
       ffffffff..................


Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N5 00 00 00 00
Aiee, Killing Interrupt handler
Kernel panic: Attempted to kill the idle task!
In swapper task - not syncing


       =[ metasploit v6.3.49-dev                          ]
+ -- --=[ 2383 exploits - 1244 auxiliary - 417 post       ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > █
```

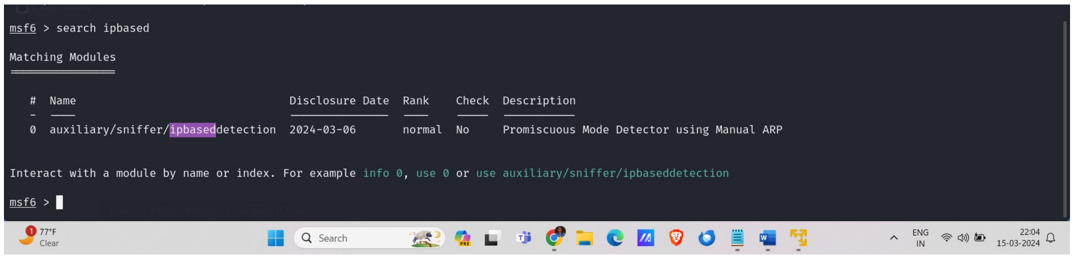
Figure 3. *msfconsole* Loading



```
┌──(root㉿kali)-[/usr/share/metasploit-framework/modules/auxiliary]
└─# msfconsole -q
Detecting sniffers ...
msf6 > reload_all
[*] Reloading modules from all module paths ...
Detecting sniffers ...

Unable to handle kernel NULL pointer dereference at virtual address 0×d34db33f
EFLAGS: 00010046
eax: 00000001 ebx: f77c8c00 ecx: 00000000 edx: f77f0001
esi: 803bf014 edi: 8023c755 ebp: 80237f84 esp: 80237f60
ds: 0018   es: 0018  ss: 0018
Process Swapper (Pid: 0, process nr: 0, stackpage=80377000)
```

Figure 4. Reloading the Scripts

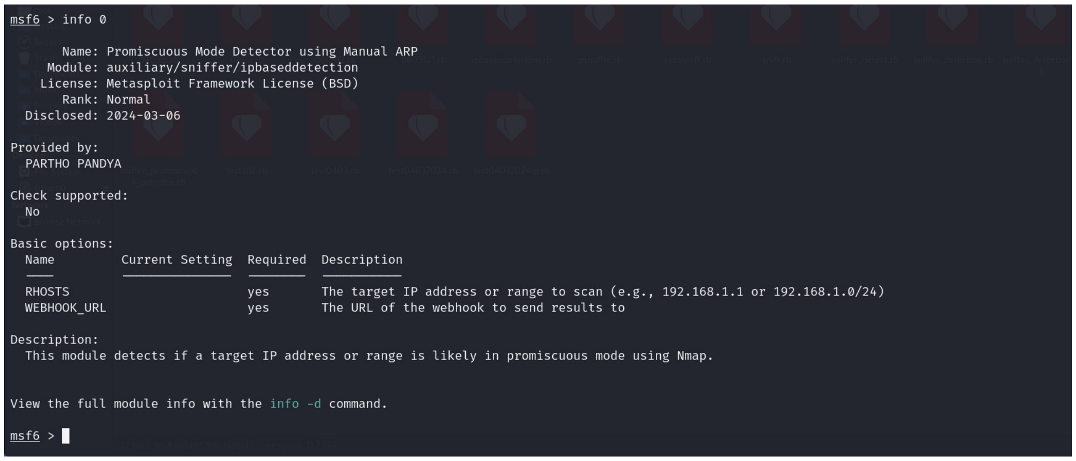
3) Next, search for the newly added module using the command "search ipbased."



```
msf6 > search ipbased

Matching Modules
================

   #  Name                                Disclosure Date  Rank    Check  Description
   -  ----                                ---------------  ----    -----  -----------
   0  auxiliary/sniffer/ipbaseddetection  2024-03-06       normal  No     Promiscuous Mode Detector using Manual ARP


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/sniffer/ipbaseddetection

msf6 > █
```

Figure 5. Module Finding Using Search

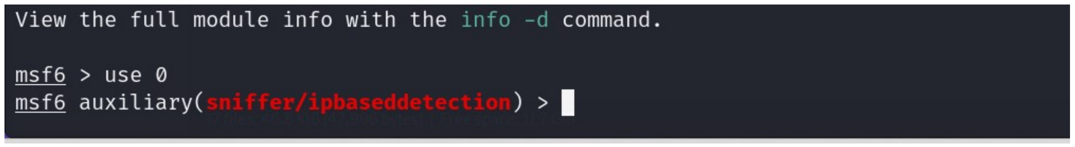
4) First, check the INFO using “Info 0”



```
msf6 > info 0

       Name: Promiscuous Mode Detector using Manual ARP
     Module: auxiliary/sniffer/ipbaseddetection
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2024-03-06

Provided by:
  PARTHO PANDYA

Check supported:
  No

Basic options:
  Name         Current Setting  Required  Description
  ----         ---------------  --------  -----------
  RHOSTS                        yes       The target IP address or range to scan (e.g., 192.168.1.1 or 192.168.1.0/24)
  WEBHOOK_URL                   yes       The URL of the webhook to send results to

Description:
  This module detects if a target IP address or range is likely in promiscuous mode using Nmap.


View the full module info with the info -d command.

msf6 >
```

Figure 6. Usage and Information

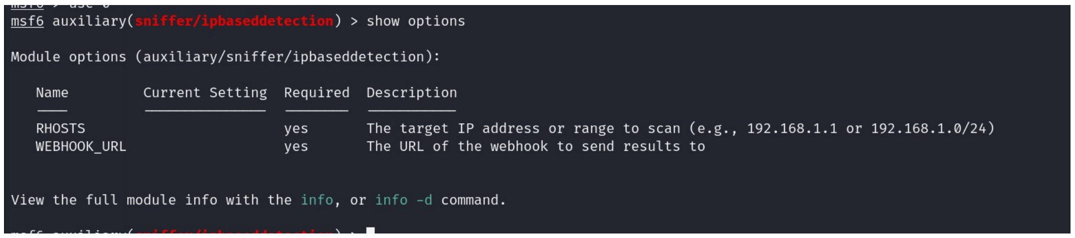
5) Now, use the command "use 0" to activate the module.



```
View the full module info with the info -d command.

msf6 > use 0
msf6 auxiliary(sniffer/ipbaseddetection) >
```

Figure 7. Using of Module

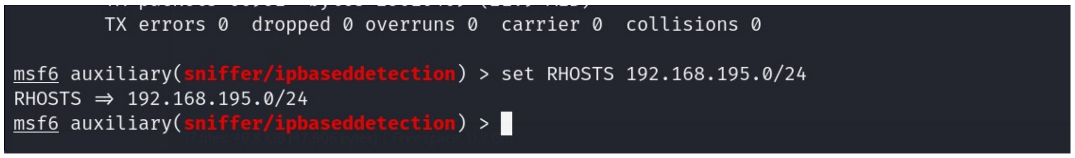
6) Now, enter the command "show options" to display the required options needed for the detection process.



```
msf6 auxiliary(sniffer/ipbaseddetection) > show options

Module options (auxiliary/sniffer/ipbaseddetection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   RHOSTS                        yes       The target IP address or range to scan (e.g., 192.168.1.1 or 192.168.1.0/24)
   WEBHOOK_URL                   yes       The URL of the webhook to send results to


View the full module info with the info, or info -d command.
```

Figure 8. Options Fetching

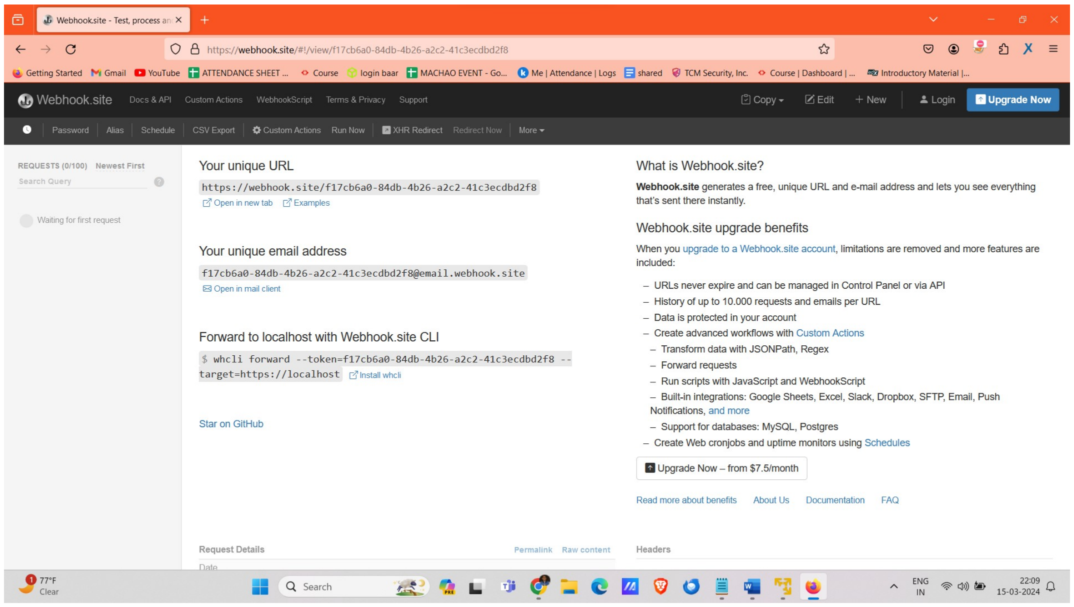
7) Now, to scan the entire IP range, you can specify the range according to the subnet. Provide the appropriate IP range.



```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

msf6 auxiliary(sniffer/ipbaseddetection) > set RHOSTS 192.168.195.0/24
RHOSTS => 192.168.195.0/24
msf6 auxiliary(sniffer/ipbaseddetection) >
```

Figure 9. Setting Target

8) Open the webhook site.



Your unique URL

```
https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
```

Open in new tab Examples

Your unique email address

```
f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8@email.webhook.site
```

Open in mail client

Forward to localhost with Webhook.site CLI

```
$ whcli forward --token=f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8 --
target=https://localhost
```

Install whcli

Star on GitHub

What is Webhook.site?

Webhook.site generates a free, unique URL and e-mail address and lets you see everything that's sent there instantly.

Webhook.site upgrade benefits

When you upgrade to a Webhook.site account, limitations are removed and more features are included:

- URLs never expire and can be managed in Control Panel or via API
- History of up to 10.000 requests and emails per URL
- Data is protected in your account
- Create advanced workflows with Custom Actions
  - Transform data with JSONPath, Regex
  - Forward requests
  - Run scripts with JavaScript and WebhookScript
  - Built-in integrations: Google Sheets, Excel, Slack, Dropbox, SFTP, Email, Push Notifications, and more
  - Support for databases: MySQL, Postgres
- Create Web cronjobs and uptime monitors using Schedules

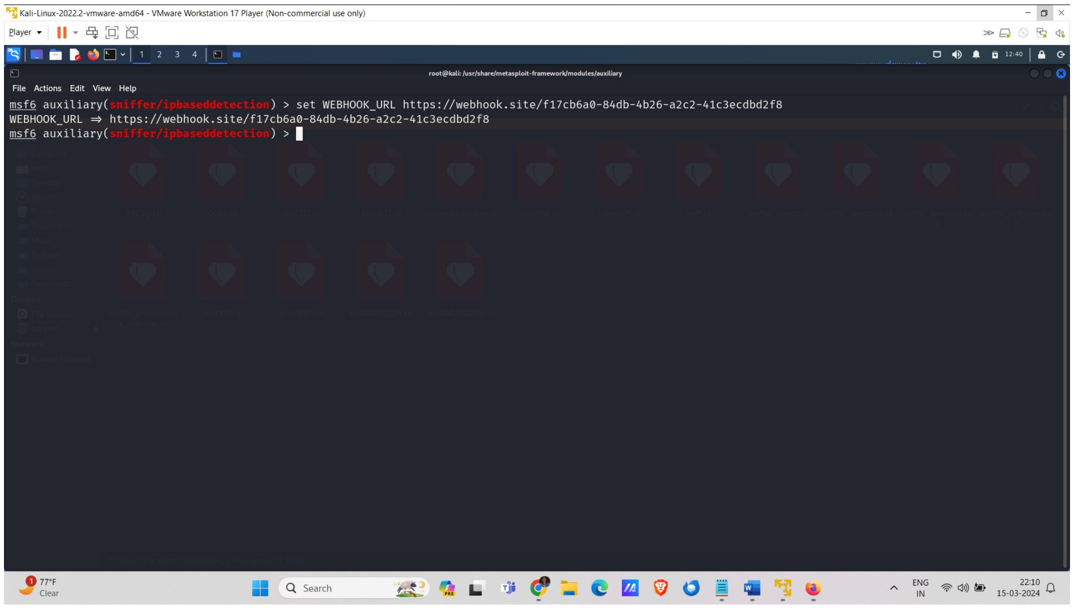
Upgrade Now – from $7.5/month

Read more about benefits About Us Documentation FAQ

Request Details Permalink Raw content Headers

Figure 10. The Party Webhook to Get the Data

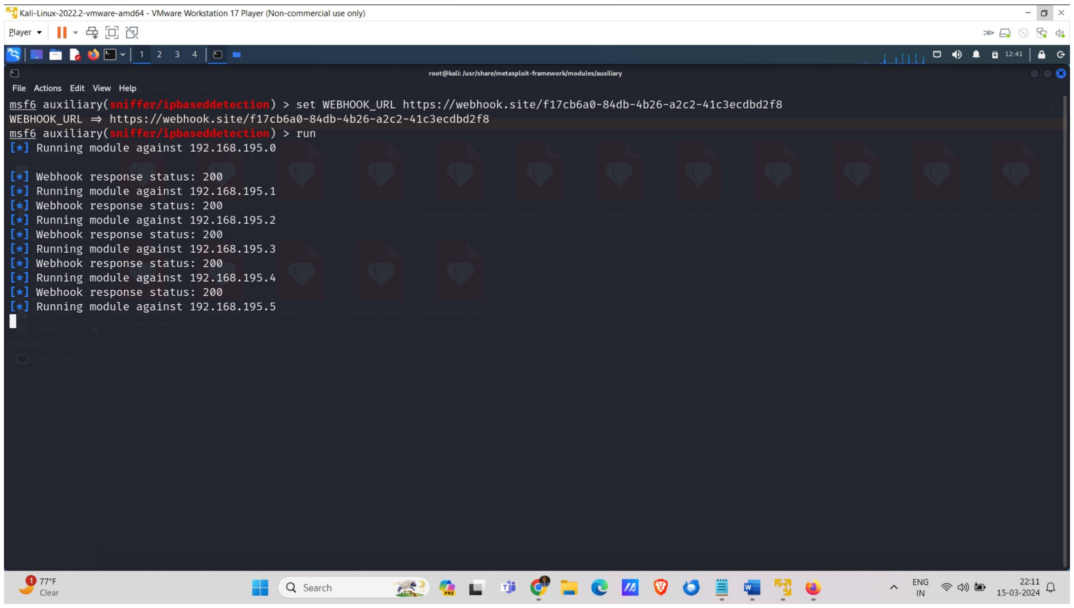
9) Copy the webhook URL and then set it as `WEBHOOK_URL` using the command `set WEBHOOK_URL "url".


Kali-Linux-2022.2-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
root@kali: /usr/share/metasploit-framework/modules/auxiliary
File Actions Edit View Help

```
msf6 auxiliary(sniffer/ipbaseddetection) > set WEBHOOK_URL https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
WEBHOOK_URL => https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
msf6 auxiliary(sniffer/ipbaseddetection) > 
```

Figure 11. Setting Up the Third-Party Location

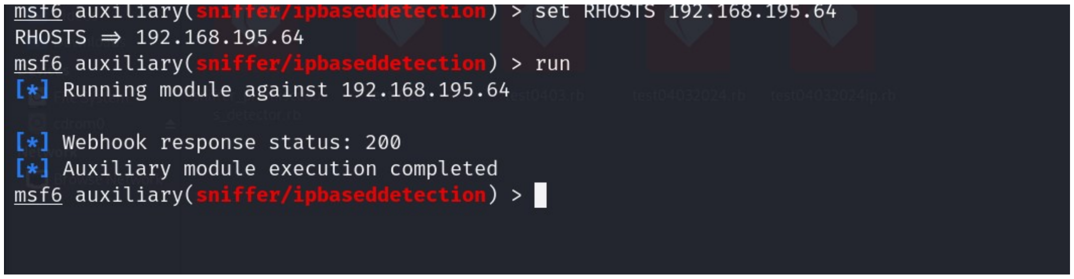
10) Now it is time to run the detection module.



```
msf6 auxiliary(sniffer/ipbaseddetection) > set WEBHOOK_URL https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
WEBHOOK_URL ⇒ https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
msf6 auxiliary(sniffer/ipbaseddetection) > run
[*] Running module against 192.168.195.0

[*] Webhook response status: 200
[*] Running module against 192.168.195.1
[*] Webhook response status: 200
[*] Running module against 192.168.195.2
[*] Webhook response status: 200
[*] Running module against 192.168.195.3
[*] Webhook response status: 200
[*] Running module against 192.168.195.4
[*] Webhook response status: 200
[*] Running module against 192.168.195.5
```

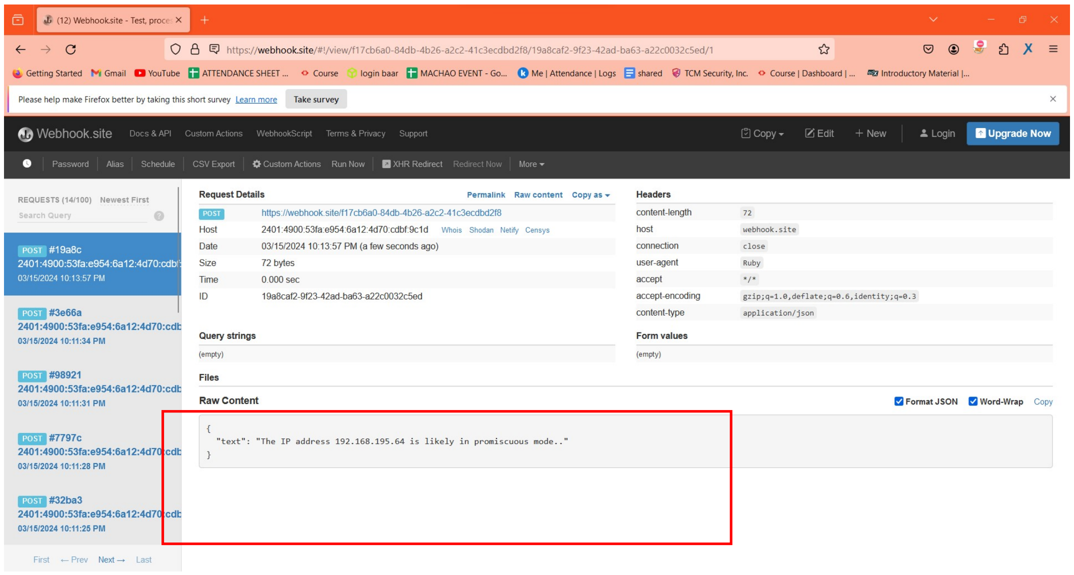
Figure 12. Running the Module

11) At the webhook site, you will see an alert indicating that promiscuous mode has been detected.



```
msf6 auxiliary(sniffer/ipbaseddetection) > set RHOSTS 192.168.195.64
RHOSTS ⇒ 192.168.195.64
msf6 auxiliary(sniffer/ipbaseddetection) > run
[*] Running module against 192.168.195.64

[*] Webhook response status: 200
[*] Auxiliary module execution completed
msf6 auxiliary(sniffer/ipbaseddetection) > 
```

Figure 13. Result Fetching



Request Details

POST	https://webhook.site/f17cb6a0-84db-4b26-a2c2-41c3ecdbd2f8
Host	2401:4900:53fa:e954:6a12:4d70:cdbf:9c1d
Date	03/15/2024 10:13:57 PM (a few seconds ago)
Size	72 bytes
Time	0.000 sec
ID	19a8caf2-9f23-42ad-ba63-a22c0032c5ed

Headers

content-length	72
host	webhook.site
connection	close
user-agent	Ruby
accept	*/*
accept-encoding	gzip;q=1.0,deflate;q=0.6,identity;q=0.3
content-type	application/json

Query strings: (empty)
Form values: (empty)

Raw Content

```
{
  "text": "The IP address 192.168.195.64 is likely in promiscuous mode.."
}
```

Figure 14. Result on Webhook

4.1.2 Accuracy Chart

The statistics for various sniffer applications, tested with 25 devices using a Windows-based setup, are presented in the figure below. Additionally, the detection accuracy for promiscuous mode in Windows is 97.37%, as discussed in the Results section. Furthermore, the detection percentage is 99.20% accurate in Linux for the promiscuous mode, as mentioned in the Results.

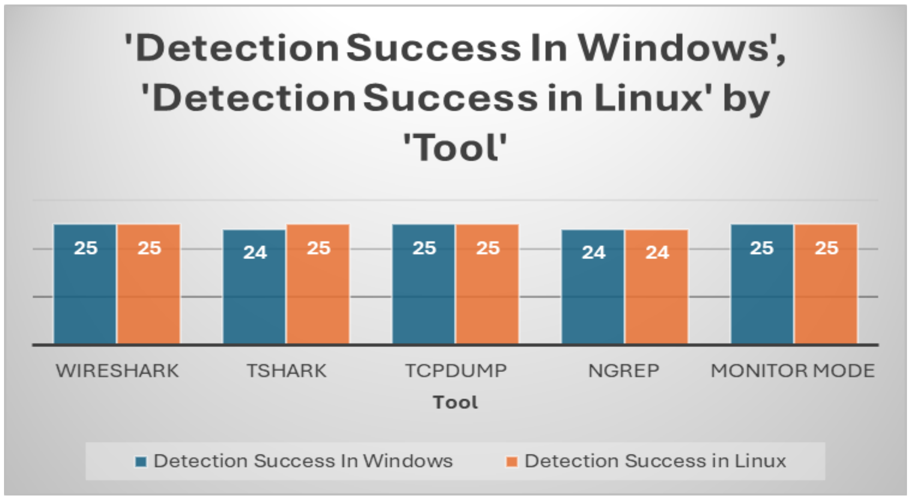


Figure 15. Dectection Chart

Table 1. Promiscuous Mode Detection Accuracy Across Tools and OS

No.	Tool	Detection Success in Windows	Detection Success in Linux	Total Test Performed
1	Wireshark	25	25	25
2	Tshark	24	25	25
3	tcpdump	25	25	25
4	ngrep	24	24	25
5	Monitor Mode	25	25	25
	Total	121	124	125
	Success percentage	97.37%	99.20%	98.25
		In windows we obtained partial detection in certain device So there is variation in percentage		

4.2. Discussion

Using the IP-based detection module added to the sniffer category by Partho Pandya in the Metasploit Framework, network administrators and penetration testing experts can ascertain the state of the NIC and its packet capturing and sniffing capabilities. This Ruby script, utilizing PACKET FU, sends specifically crafted packets to query the network interface card's status, checking for promiscuous mode conditions. An alert generated on the webhook indicates the NIC's status. The script sends these packets using OSI reference model layers 2-3 to check the NIC's status. Results obtained from multiple devices and statistical analysis demonstrate the script's effectiveness in detecting monitor mode, also known as promiscuous mode.

The script was tested on 25 devices, detecting 24 devices using Wireshark, 24 devices using tcpdump (with 1 device not alerting on the webhook), 24 devices using ngrep, and all 25 devices using monitor mode and t-shark. The statistics provided in this dissertation indicate that the script is highly effective in detecting the NIC's status, aiding in identifying devices passively sniffing the network.

According to the literature, no existing work utilizes the specific modules employed in this study; previous efforts have been conducted in other languages, and no comparable module exists in Metasploit for this purpose. With a detection accuracy of 97.37% in Windows-based setups and 99.20% in Linux-based setups, the script demonstrates an overall detection accuracy of 98.285%. It is effective and can be further improved for future applications. The instances of non-detection were primarily due to older Raspberry Pi setups with outdated NIC wireless interfaces, which occasionally resulted in false negatives.

5. Conclusion

This research concludes that passive detection plays a crucial role in network security through the use of the provided script. The script can be automated using Crontab or Cronjob commands, eliminating the need for manual execution. Once the script is set up on the device connected to the network interface, a few commands will automate the entire function in the background, with the output directed to `/dev/null`. This ensures seamless background operation without requiring user intervention.

With a detection accuracy of 98.285%, the script demonstrates significant potential for future enhancements. A potential future implementation could involve developing a GUI-based Android application capable of directly controlling the router and blocking specific IP or MAC addresses using appropriate methods. This application could be deployed on a cloud server or as an IoT device IPS, running the script or module to perform passive detection effectively.

Additionally, future work could incorporate secure telnet to enable remote administration of the router or network, optimizing the detection and security capabilities. This advancement would provide a robust solution for managing and securing networks against passive sniffing and other malicious activities.

Acknowledgment

At this significant juncture of my academic journey, I wish to take this opportunity to express my heartfelt gratitude to all those who have supported, guided, and inspired me throughout the completion of this dissertation.

- I am deeply thankful to Dr. Kapil Kumar, Head of Department and my project guide, for his invaluable guidance and for believing in my potential. His trust, encouragement, and constructive feedback played a vital role in shaping this work and giving it meaningful direction.
- I extend my sincere thanks to my co-guide, Mr. Kashyap Joshi, whose unwavering support and insightful mentorship were instrumental throughout this project. His clarity of vision, timely suggestions, and patient guidance helped me turn my ideas into tangible outcomes.
- I also wish to express my gratitude to Decimal Point analytics, where I had the opportunity to work as a Senior cyber security engineer. The hands-on exposure and industry experience I gained there allowed me to bridge the gap between academic concepts and practical application ultimately helping me refine and finalize the subject of this dissertation.

Above all, I owe everything to the unconditional love and support of my parents. My mother, Shital Pandya, has been the silent strength behind all my efforts, constantly motivating me to strive for excellence. Her belief in me and her encouragement to pursue a career in cybersecurity have been foundational to my achievements. I am equally grateful to my father, Alkesh Pandya, whose calm presence and unwavering support provided the strength I needed to overcome challenges along the way.

To each of them, I extend my deepest gratitude.

References

[1] G. S. Rao and P. K. Subbarao, “A novel framework for detection of DoS/DDoS attack using deep learning techniques, and an approach to mitigate the impact of DoS/DDoS attack in network environment,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 1, pp. 450–466, 2024.

[2] F. S. Alsharbaty and Q. I. Ali, “Smart electrical substation cybersecurity model based on WPA3 and cooperative hybrid intrusion detection system (CHIDS),” *International Journal of System Assurance Engineering and Management*, vol. 15, no. 2, pp. 389–402, 2024.

[3] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, “Hybrid deep learning-based model for anomaly detection in IoT networks,” *Computers & Security*, vol. 123, Art. no. 102927, pp. 1–14, 2022.

[4] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, “Server-side detection of content sniffing attacks,” in *Proc. IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp. 193–202, 2011.

[5] S. R. Kumbhare and A. S. Shirsat, “Agent based network sniffer detection system,” *International Journal of Scientific and Research Publications*, vol. 3, no. 4, pp. 1–5, Apr. 2013.

[6] S. Behl and A. Behl, “An approach to detect packets using packet sniffing,” *International Journal of Computer Science and Engineering Survey*, vol. 4, no. 3, pp. 21–30, 2013.

[7] M. Al-Saleh and K. Al-Sarayreh, “An intelligent approach of sniffer detection,” *The International Arab Journal of Information Technology*, vol. 9, no. 1, pp. 45–52, Jan. 2012.

[8] R. Verma and A. Singh, “Experimental and comparative analysis of packet sniffing tools,” in *Advances in Intelligent Systems and Computing*, vol. 800, pp. 567–576, 2019.

[9] A. A. Yassin, A. H. Mustafa, and M. A. Al-Dabbagh, “Real-world ARP attacks and packet sniffing: Detection and prevention on Windows and Android devices,” in *Proc. International Conference for Informatics and Information Technology (CIIT)*, pp. 89–94, 2015.

[10] P. Sharma and R. Gupta, “Sniffing: A major threat to secure socket layer and its detection,” in *Proc. CSI International Conference on Computer Communication and Networks (CSI-COMNET)*, pp. 112–117, 2011.

[11] T. Hasegawa, Y. Shinoda, and K. Okada, “Malicious sniffing systems detection platform,” in *Proc. International Symposium on Applications and the Internet (SAINT’04)*, pp. 120–127, 2004.

[12] F. Guo and T. Chiueh, “Detection of sniffers in an Ethernet network,” in *Lecture Notes in Computer Science*, vol. 3042, pp. 213–226, 2004.

[13] R. Spangler, “Packet sniffer detection with AntiSniff,” Foundstone Inc., Technical Report, pp. 1–18, 2003.

[14] S. Kumar and R. Kumar, “Comparative study of two most popular packet sniffing tools—Tcpdump and Wireshark,” in *Proc. International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 95–100, 2017.

[15] A. Patel and N. Shah, “Packet sniffing: Monitoring and analysis using sniffing method,” *Journal of Emerging Technologies and Innovative Research*, vol. 5, no. 11, pp. 345–350, Nov. 2018.

[16] Y. Chen, W. Trappe, and R. P. Martin, “A location-aware rogue AP detection system based on wireless packet sniffing of sensor APs,” in *Proc. ACM Symposium on Applied Computing (SAC)*, pp. 1134–1139, 2011.

[17] S. R. Jangra and V. Jain, “Network traffic analysis and intrusion detection using packet sniffer,” in *Proc. International Conference on Computer and Communication Software Networks (ICCSN)*, pp. 410–414, 2010.

[18] S. A. Al-Janabi and H. M. Salman, “Packet sniffing and sniffing detection,” *International Journal of Information Engineering and Technology*, vol. 1, no. 4, pp. 301–306, 2009.

[19] J. Li, Z. Zhao, and R. Li, “Sniffing detection based on network traffic probing and machine learning,” *IEEE Access*, vol. 8, pp. 152340–152351, 2020.

[20] H. Zhang, G. Cheng, and P. Dong, “Robust and efficient Wi-Fi traffic classification with deep learning,” *Computer Networks*, vol. 222, Art. no. 109559, pp. 1–13, 2023.

[21] M. A. Khan and S. Salahuddin, “A lightweight anomaly detection system for black hole attack,” *Electronics*, vol. 12, no. 6, pp. 1294–1307, 2023.

[22] T. Phan and J. Park, “Efficient distributed denial-of-service attack defense in SDN-based cloud,” *IEEE Access*, vol. 7, pp. 18701–18714, 2019.

[23] K. Sambangi and N. Gondi, “A machine learning approach for DDoS attack detection using multiple linear regression,” in *Proc. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1049–1054, 2020.

[24] B. Jia, X. Huang, R. Liu, and Y. Ma, “A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning,” *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–9, 2017.

[25] L. M. Ibrahim, “Anomaly network intrusion detection system based on distributed time-delay neural network,” *Journal of Engineering Science and Technology*, vol. 5, no. 4, pp. 457–471, 2010.

[26] A. Perez-Diaz, A. Valdovinos, K.-K. R. Choo, and D. Zhu, “A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning,” *IEEE Access*, vol. 8, pp. 155859–155872, 2020.
[27] A. Kuzmanovic and E. W. Knightly, “Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants,” in *Proc. ACM SIGCOMM*, pp. 75–86, 2003.
[28] J. Jensen and N. Gruschka, “SWAP: Mitigating XSS attacks using a reverse proxy,” Technical Report, 2014.
[29] T. King, “Packet sniffing in a switched environment,” SANS Institute, GSEC Practical, ver. 1.4, pp. 1–32, 2006.
[30] S. Saroha, “Restraining packet sniffing and security,” Technical Report, pp. 1–10, 2012.