OCTOBER 26, 2024

Network Packet Analyzer

Real-Time Network Traffic Analysis Using Python and Scapy

JAGADISH TRIPATHY

Jagadishtripathy144@gmail.com

Table of Contents

Network Packet Analyzer

1. Executive Summary

This report outlines the design and implementation of a **Network Packet Analyzer** tool developed in Python using the Scapy library. The tool captures live network traffic from a designated network interface, providing essential information about each packet, such as source and destination IP addresses, MAC addresses, protocols, and payload data. This project is intended for educational purposes and is restricted to controlled environments, ensuring ethical usage.

The tool serves as an introduction to network traffic analysis and lays a foundation for further research into packet inspection, protocol dissection, and security monitoring.

2. Objectives

1. **Develop a packet sniffer** capable of capturing real-time network packets from a specified interface.
2. **Analyze captured packets** to display vital information, including IP and MAC addresses, protocols (TCP, UDP, ICMP), and payload data.
3. **Enhance understanding** of network protocols and traffic flow through hands-on implementation.
4. **Ensure ethical use** of the tool strictly within permitted environments for educational purposes.

3. Methodology

3.1 Tool Design

The packet analyzer is written in Python and utilizes the **Scapy** library to capture and analyze packets. Scapy's versatility allows for multi-protocol support and easy packet dissection, essential for extracting detailed packet-level information.

3.2 Environment Setup

- **Initialization**: The program is executed with root privileges (*sudo*) to enable network sniffing.
- **Packet Filtering**: To capture relevant network packets, filters are applied for IP, TCP, UDP, and ICMP protocols, providing a broad yet meaningful dataset for analysis.
- **Packet Analysis**: Each captured packet is parsed, and essential information such as source/destination IP and MAC addresses, protocols, and payloads is displayed.

3.4 Ethical Considerations

The packet analyzer tool was deployed exclusively in a secure, controlled environment to comply with cybersecurity and privacy regulations. Unauthorized packet sniffing constitutes a breach of privacy and legal statutes.

4. Findings

4.1 Packet Details Captured

The analyzer successfully captured a variety of packet types, including:

- **TCP Packets**: Observed between local devices and external IPs during web browsing and SSH connections.
- **UDP Packets**: Noted during DNS resolution and DHCP processes.
- **ICMP Packets**: Captured during network diagnostics, such as ping tests.

4.2 Observed Traffic Patterns

- **ARP Broadcasts**: The analyzer frequently captured ARP broadcast packets indicating active devices requesting MAC addresses for IPs on the network.
- **Consistent Source IP**: When the tool was tested on a single device network, most captured traffic had the same source IP address, consistent with local traffic generation.

4.3 Limitations

- **Broadcast Traffic**: Due to network settings, a significant portion of captured packets were broadcast (ARP) packets, limiting the diversity of the data.
- **Limited Protocol Range**: While TCP, UDP, and ICMP are covered, other protocols could be incorporated to expand the analyzer's versatility.

5. Analysis & Insights

The tool provides a foundational view of network activity, highlighting commonly used protocols and interactions. Such packet-level insights are essential in fields like:

- **Network Security**: Detecting suspicious packets or unusual IPs.
- **Performance Monitoring**: Identifying high-traffic sources or potential network congestion.
- **Protocol Analysis**: Understanding protocol-specific behaviors, which is valuable for identifying vulnerabilities or misconfigurations.

The tool demonstrates that, even with basic filtering, valuable insights can be drawn from network traffic, showcasing its potential as an educational asset for beginners in cybersecurity.

6. Recommendations

6.1 Expanding Protocol Support

Implementing support for additional protocols, such as HTTP, HTTPS, DNS, and DHCP, would broaden the tool's utility in diverse network environments.

6.2 Improving Data Visualization

Integrating a data visualization tool, such as Matplotlib, could provide visual representations of traffic patterns and protocol distribution, enhancing interpretability.

6.3 Deploying in Controlled Network Segments

Testing in more diverse network environments, including those with varied devices and protocols, could yield additional insights and improve the tool’s robustness.

7. Conclusion

The Network Packet Analyzer project successfully captured and analyzed real-time network traffic, demonstrating fundamental concepts in network monitoring and analysis. By dissecting network packets, this tool offers practical experience in protocol analysis and traffic inspection. It serves as a valuable starting point for students and enthusiasts interested in network security, laying the groundwork for more sophisticated tools and analyses in cybersecurity.

Appendix: Sample Code

```
from scapy.all import sniff

from datetime import datetime

def analyze_packet(packet):

    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    src_mac = packet.src

    dst_mac = packet.dst

    print(f"[+] Packet captured at: {timestamp}")

    print(f"Source MAC: {src_mac}")

    print(f"Destination MAC: {dst_mac}")

    print("------------------------------------------------------------")

def start_sniffing(interface="eth0", packet_count=10):
```

print(f"[*] Starting packet capture on {interface}...\n")

sniff(iface=interface, prn=analyze_packet, count=packet_count, filter="ip or tcp or udp or icmp")

start_sniffing()

Full Code Link

GitHub

Contact Information

For further information or discussion on network security and analysis, please feel free to reach out via [Linkedin/ GitHub/Email].