NETWORK ANALYSIS FOR MALICIOUS PACKETS USING WIRESHARK

SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIREMENTS FOR AWARD OF THE SIX MONTHS ONLINE CERTIFICATE COURSE IN CYBER SECURITY

Course Duration: [22-01-2024 to 21-07-2024]

By K .B .M Sushanth
(2406CYS103)
Under the Esteemed Guidance
Prof.Niladri dey

INTRODUCTION

• Importance of Network Analysis

Network analysis is crucial for maintaining the security and integrity of an organization's network.It involves monitoring, capturing, and analyzing network traffic to detect anomalies and potential threats.

• Detecting Malicious Packets

Identifying malicious packets is essential to prevent unauthorized access, data breaches, and other cyber threats.Timely detection allows for quick mitigation of risks.

• Introduction to Wireshark

Wireshark is a powerful open-source network protocol analyzer that provides detailed insights into network traffic. " "It is widely used by network administrators and security professionals for troubleshooting and security analysis.

OVERVIEW OF WIRESHARK

• What is Wireshark?

Wireshark is an open-source tool for capturing and analyzing network traffic in real time. It supports a wide range of protocols and provides a user-friendly interface for in-depth analysis

• Key Features:

- Real-time capture and offline analysis
- Rich display filters
- Support for hundreds of protocols

• Importance in Network Security

Wireshark helps in identifying security vulnerabilities, detecting malicious activities, and troubleshooting network issues. " "Its comprehensive analysis capabilities make it an invaluable tool for maintaining network security.

PROJECT SCENARIO: MALICE.COM

- Example Website: malice.com

In this project, we use malice.com as a hypothetical malicious website to demonstrate network analysis and blocking techniques.

- Hypothetical Threat Scenario

Assume that malice.com is a known malicious site that attempts to exploit vulnerabilities in network devices.The goal is to detect and block any connection attempts to this site.

- Objective: Detect and Block Connections to malice.com

Using Wireshark, we will analyze network traffic, create rule engines to identify attempts to access malice.com, " "and implement these rules to block such connections.

NETWORK ANALYSIS WITH WIRESHARK

• Setting Up Wireshark

Download and install Wireshark from the official website.

Configure network interfaces to capture traffic.

• Capturing Network Traffic

- Start a new capture session to monitor network traffic

- Ensure the capture is running on the correct network segment to include all relevant traffic.

• Filtering and Analyzing Packets

- Use display filters to isolate traffic related to malice.com (e.g., http.host == \"malice.com\").

- Analyze the captured packets to identify any patterns or indicators of malicious activity.

• Identifying Malicious Activity Related to malice.com

- Look for unusual traffic patterns, repeated connection attempts, and other signs of malicious activity.

- Document findings to support rule creation.

CREATING RULE ENGINES

- Introduction to Rule Engines

Rule engines are tools used to define and apply conditions for network traffic filtering and blocking.They are crucial for automated detection and response to malicious activities.

- Importance in Network Security

By using rule engines, organizations can proactively protect their networks from known threats by blocking traffic that matches specific malicious patterns.

- Process of Creating Rules in Wireshark

Although Wireshark itself does not create blocking rules, it helps identify traffic patterns that should be blocked. The insights gained from Wireshark can be used to create rules for firewalls or intrusion detection systems.

EXAMPLE RULE ENGINE 1: BLOCKING HTTP REQUESTS

- Rule Definition:

  ```

  alert tcp any any -> any 80 (msg:"Block malice.com HTTP request"; content:"malice.com"; http_host; sid:100001;)

  ```

- Explanation:

  - Inspects HTTP traffic on port 80

  - Triggers an alert if the host is malice.com

- Impact:

  - Prevents devices from accessing malice.com via HTTP

- Enhances network security by stopping potential malicious activity at the source.

EXAMPLE RULE ENGINE 2: BLOCKING DNS REQUESTS

- Rule Definition:

  ```

  alert udp any any -> any 53 (msg:"Block malice.com DNS request"; content:"malice.com"; dns_query; sid:100002;)

  ```

- Explanation:

  - Inspects DNS traffic

  - Triggers an alert if a DNS query for malice.com is detected

- Impact:

  - Blocks DNS resolution for malice.com, preventing access

- Stops potential malicious activity by disrupting the DNS query process.

EXAMPLE RULE ENGINE 3: BLOCKING HTTPS REQUESTS

- Rule Definition:

  ```

  alert tcp any any -> any 443 (msg:"Block malice.com HTTPS request"; content:"malice.com"; tls_sni; sid:100003;)

  ```

- Explanation:

  - Inspects HTTPS traffic on port 443

  - Triggers an alert if the Server Name Indicator (SNI) is malice.com

- Impact:

  - Prevents devices from accessing malice.com via HTTPS

- Enhances network security by stopping potential encrypted malicious activity.

IMPLEMENTATION AND RESULTS

APPLYING THE RULES IN A NETWORK ENVIRONMENT

When it comes to applying rules in a network environment using Wireshark, we're essentially crafting guidelines for how our network devices should behave. Here's how this process unfolds:

Rule Definition:

Start by identifying the specific behavior you want to regulate. For example, you might want to block all incoming traffic from a certain IP address range or restrict access to specific services (e.g., blocking FTP traffic).

Create firewall rules based on your requirements. These rules can be simple (e.g., allow/deny based on IP addresses) or more complex (e.g., deep packet inspection rules).

Firewall Configuration:

Implement these rules on your network devices (firewalls, routers, switches). Wireshark itself doesn't directly block traffic; it provides insights for creating effective rules.

Consider both inbound and outbound rules. For example:

Inbound: Block incoming traffic from known malicious IP addresses.

Outbound: Prevent devices on your network from connecting to suspicious domains (like our fictional "malice.com").

Testing the Rules:

Before deploying rules in a production environment, test them in a controlled setting.

Use Wireshark to capture network traffic while applying the rules. Verify that the expected behavior occurs—malicious traffic is blocked, and legitimate traffic passes through.

• TESTING AND VERIFYING THE RULES

1. Wireshark as Your Detective:
    1. Capture live network data using Wireshark. This involves selecting the appropriate network interface and starting the capture process.
    2. Use display filters to focus on relevant packets. For example, filter by protocol (e.g., HTTP, DNS) or IP addresses.
    3. Analyze captured packets to ensure they align with your rule expectations.

2. Coloring Rules:
    1. Wireshark color-codes packets based on various criteria (e.g., protocol, error conditions).
    2. Customize coloring rules to highlight specific patterns. Testing While Capturing:
    3. While a capture is running, you can:
        1. Stop the capture: When you've collected enough data.
        2. Restart a running capture: Useful for fine-tuning your rules.

EFFECTIVENESS OF THE RULES IN BLOCKING MALICIOUS TRAFFIC

1. Validation:
    1. After deploying your rules, monitor network traffic.
    2. Check if the expected malicious traffic is indeed blocked. Look for signs of attempted connections to blacklisted IPs or domains.
2. Fine-Tuning:
    1. Adjust rules as needed. Sometimes false positives occur, blocking legitimate traffic. Refine rules based on real-world observations.
3. Continuous Monitoring:
    1. Network threats evolve. Regularly review and update your rules to stay effective.