SCIENCE
PASSION
TECHNOLOGY

Real-time Network Traffic Analysis and Feature Extraction

Riccardo Baljak, Ismar Nurdinović, Javier Nieto Castaño

January, 2025

Network traffic analysis

Capturing, inspecting, and interpreting data packets transmitted over a network. [Bar+20]

Network traffic analysis

Network traffic analysis

- Flow Analysis

Network traffic analysis

- Flow Analysis
- Performance Monitoring

Network traffic analysis

- Flow Analysis
- Performance Monitoring
- ...

Network traffic analysis

- Flow Analysis
- Performance Monitoring
- ...
- Packet Capture and Inspection

What are we doing?

Explaining the importance, and showing real-world examples of Network Traffic Analysis and feature extraction

Why are we doing it?

Show and explain the importance of real-time security in today's increasing network usage

Importance

- Increasing complexity: Cloud, IoT, and 5G.
- Cyber threats like DDoS [DM04] and APTs [DM04] require real-time response.
- Real-time analysis reduces risks [Som+17].

Key Concepts

- **Packet Capture:**
  - Involves capturing and inspecting data packets transmitted over a network.
  - Packets contain headers and payloads with information about source, destination, and protocol.

Key Concepts

- **Anomaly Detection:**
  - Identifies unusual patterns in network traffic that may indicate security threats.
  - Utilizes machine learning and statistical techniques to detect anomalies in real-time.

Pseudo-Code

This pseudo-code is a very simplified representation of the Python source code provided (without anomaly detection):

Pseudo-Code

This pseudo-code is a very simplified representation of the Python source code provided (without anomaly detection):

Input: interface, packet_count
Output: Formatted packet data
pcap_reader ← LiveCapture(*interface*)
while *not* KeyboardInterrupt **do**
    foreach *packet* from *pcap_reader.sniff_continuously(packet_count)* **do**
        print *packet number* and *timestamp*
        if *'eth'* ∈ *packet* **then**
            print *source* and *destination MAC address*
        end
        if *'ip'* ∈ *packet* **then**
            print *IP version number*, *protocol type*, *protocol number*, *source* and *destination IP address*
        end
        if *'tcp'* ∈ *packet* **then**
            print *source* and *destination port*, *flags*, *window size* and *checksum*
        end
        if *'udp'* ∈ *packet* **then**
            print *source* and *destination port*, *length* and *checksum*
        end
    end
end

Explanation

Explanation

1. Initialize the `pcap (= packet capture) reader` to be a `LiveCapture` of the desired `interface`.

Explanation

1. Initialize the `pcap (= packet capture) reader` to be a `LiveCapture` of the desired `interface`.
2. Go though every sniffed packet (until the threshold is reached).

Explanation

1. Initialize the `pcap (= packet capture) reader` to be a `LiveCapture` of the desired `interface`.
2. Go though every sniffed packet (until the threshold is reached).
3. Print the basic information (packet number and timestamp).

Explanation

4. Check for distinct metrics of the packet and print different information:

Explanation

4. Check for distinct metrics of the packet and print different information:
   a. Ethernet: source and destination `MAC` address

Explanation

4. Check for distinct metrics of the packet and print different information:
   a. Ethernet: source and destination `MAC` address
   b. IP: IP version number, protocol type, protocol number, source and destination IP address

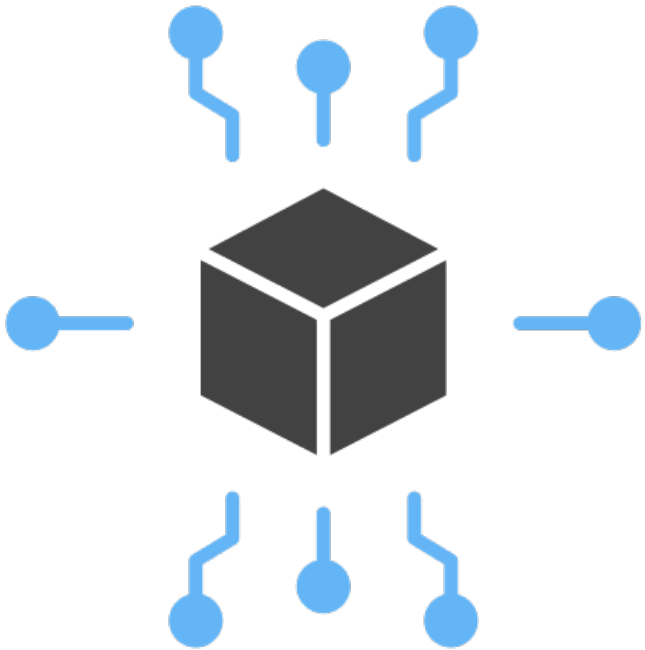
Explanation

4. Check for distinct metrics of the packet and print different information:
   a. Ethernet: source and destination `MAC` address
   b. IP: IP version number, protocol type, protocol number, source and destination IP address
   c. TCP: source and destination port, flags, window size and checksum

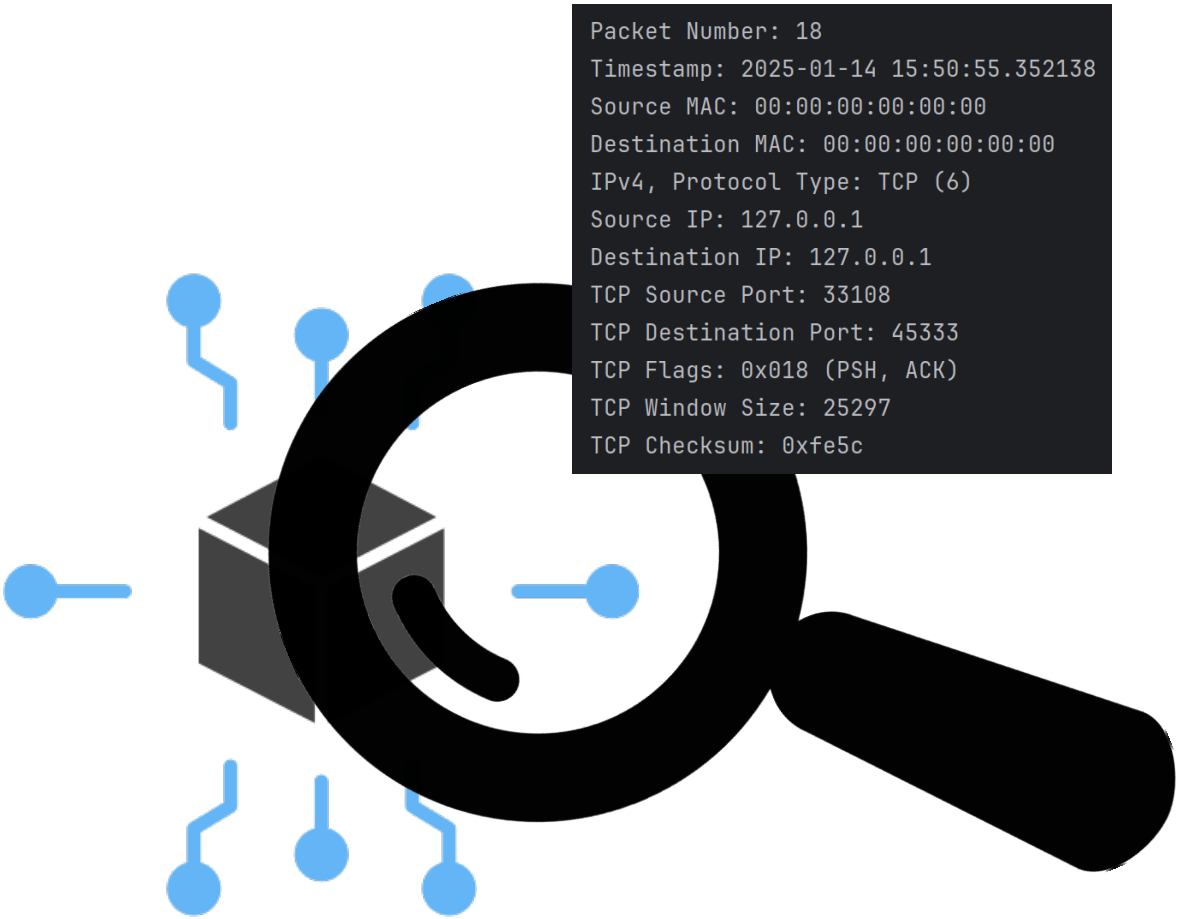
Explanation

4. Check for distinct metrics of the packet and print different information:
   a. Ethernet: source and destination `MAC` address
   b. IP: IP version number, protocol type, protocol number, source and destination IP address
   c. TCP: source and destination port, flags, window size and checksum
   d. UDP: source and destination port, length and checksum

Example



Example


Packet Number: 18
Timestamp: 2025-01-14 15:50:55.352138
Source MAC: 00:00:00:00:00:00
Destination MAC: 00:00:00:00:00:00
IPv4, Protocol Type: TCP (6)
Source IP: 127.0.0.1
Destination IP: 127.0.0.1
TCP Source Port: 33108
TCP Destination Port: 45333
TCP Flags: 0x018 (PSH, ACK)
TCP Window Size: 25297
TCP Checksum: 0xfe5c

Getting the Malicious IPs

The next code simplifies the filters that we have applied.

Getting the Malicious IPs

The next code simplifies the filters that we have applied.

Input: IP address *ip*
Output: **True** if *ip* is malicious, **False** otherwise
Initialize *headers* $\leftarrow$ {'Accept': 'application/json', 'Key': API_KEY};
Initialize *params* $\leftarrow$ {'ipAddress': ip, 'maxAgeInDays': 90};
Send HTTP GET request to *ABUSEIPDB_URL* with *headers* and *params*;
if *response is successful* **then**
    abuse_score $\leftarrow$ *response.json*()['*data*']['*abuseConfidenceScore*'];
    if *abuse_score* $>$ 50 **then**
        Print: "Malicious IP: *ip* (Score: *abuse_score*)";
        return True;
    end
end
else
    Print: "Error querying *ip*";
end
return False;

Explanation

Explanation

1. Connect to the API of a webpage called AbuseIPDB

Explanation

1. Connect to the API of a webpage called AbuseIPDB
2. For every IP we get from the packets we check it

Explanation

1. Connect to the API of a webpage called AbuseIPDB
2. For every IP we get from the packets we check it
3. Is it is flag as malicious we send a message

Explanation

1. Connect to the API of a webpage called AbuseIPDB
2. For every IP we get from the packets we check it
3. Is it is flag as malicious we send a message
4. Otherwise we check the next IP

Filters

Filters

Input: Packet p
Output: **True** if p is anomalous, **False** otherwise
`Check Packet Size`;
Extract *length* from p;
if $length > 1500 \vee length < 50$ **then**
    Log: "Unusual packet size";
    return True;
end

Input: Packet p
Output: **True** if suspicious DNS, **False** otherwise
if *'dns'* $\in p$ **then**
    foreach *part in p.dns.qry_name.split('.')* **do**
        if *Length(part)* $\geq 10 \wedge$ *Vowels(part)* ≤ 2 **then**
            return True (Suspicious domain);
        end
    end
    if *EndsWith(qry_name, ".xyz")* $\vee$ *Length(qry_name)* > 20 **then**
        return True (Unusual domain);
    end
end

Input: Packet p, Threat API T
Output: **True** if p contains malicious IP, **False** otherwise
Extract *src_ip*, *dst_ip* from p;
abuse_score $\leftarrow T.query(src\_ip)$;
if *abuse_score* > 50 **then**
    Log: "Dynamic Malicious Source IP";
    return True;
end
abuse_score $\leftarrow T.query(dst\_ip)$;
if *abuse_score* > 50 **then**
    Log: "Dynamic Malicious Destination IP";
    return True;
end

Input: Packet p, Threshold T_p
Output: **True** if p is anomalous, **False** otherwise
Track *src_ip* $\rightarrow$ *dst_port*;
if *Unique dst_port for src_ip* $> T_p$ **then**
    Log: "Potential Port Scan";
    return True;
end

Explanation

Explanation

We created some basic example filters

Explanation

We created some basic example filters

1. Check if the packet size is too high or too low

Explanation

We created some basic example filters

1. Check if the packet size is too high or too low
2. Check if the DNS has a strange pattern

Explanation

We created some basic example filters

1. Check if the packet size is too high or too low
2. Check if the DNS has a strange pattern
3. Check if an IP could be malicious using the function from before

Explanation

We created some basic example filters

1. Check if the packet size is too high or too low
2. Check if the DNS has a strange pattern
3. Check if an IP could be malicious using the function from before
4. Check if the packet IP tries to connect to many ports

Conclusion

- Real-time network traffic analysis is crucial for addressing modern cybersecurity challenges.
- Combining advanced techniques ensures efficient and timely threat detection.
- Continuous technology advancements means we need to continuously advance in security measures as well.

Questions?

Thanks for the attention!

[Bar+20] M. Barabas et al. **Real-time network traffic monitoring and analysis: A survey.** *Journal of Network and Computer Applications* (2020).

[DM04] C. Douligeris and A. Mitrokotsa. **DDoS attacks and defense mechanisms: classification and state-of-the-art.** *Computer Networks* (2004).

[Som+17] G. Somani et al. **DDoS mitigation techniques in cloud computing: Challenges and opportunities.** *IEEE Communications Surveys & Tutorials* (2017).

All icons used are from https://thenounproject.com/ and are licensed under the *Creative Commons* license.