Q1

1. **Network-Based IDS (NIDS):**
    - **Usage:** Monitors network traffic in real-time and identifies suspicious patterns or anomalies.
    - **Circumstances:** Ideal for large-scale networks where monitoring at the perimeter is crucial.
    - **Capabilities:** Analyzes packets and identifies malicious activity.

2. **Host-Based IDS (HIDS):**
    - **Usage:** Monitors activities on individual computers or hosts, looking for suspicious behavior or changes.
    - **Circumstances:** Suitable for environments where internal threats are a concern or for critical servers that require close monitoring.
    - **Capabilities:** Analyzes system logs, file integrity, and other host-specific data to detect abnormal activities.

3. **Signature-Based IDS/IPS:**
    - **Usage:** Matches known patterns or signatures of known threats.
    - **Circumstances:** Effective against known attack types with well-defined signatures.
    - **Capabilities:** Compares network traffic or system activity against a database of known attack signatures and triggers an alert or takes action if a match is found.

4. **Anomaly-Based IDS/IPS:**
    - **Usage:** Identifies deviations from normal behavior patterns, rather than relying on predefined signatures.
    - **Circumstances:** Useful for detecting previously unknown threats or variations in attack patterns.
    - **Capabilities:** Learns what "normal" behavior looks like and raises an alert or takes action when significant deviations are detected.

Q2

1. **Sniffer Mode:**

In Sniffer mode, Snort functions as a passive network sniffer. It analyzes packets on the network without actively blocking or modifying them.

2. **Packet Logger Mode:**

Packet Logger mode extends the capabilities of Sniffer mode by logging captured packets to disk for later analysis. In this mode, Snort records packet data to log files, enabling administrators to review and investigate network activity over an extended period.

3. **Network Intrusion Detection System (NIDS) Mode:**

NIDS mode is the most active and widely used operational mode of Snort. In this mode, Snort actively analyzes network traffic, compares it against predefined rules or signatures, and generates alerts or takes actions when suspicious or malicious activity is detected. NIDS mode allows Snort to function as a real-time intrusion detection system.

Q3

1. **Traditional Rules**

Snort's intrusion detection and prevention system relies on the presence of Snort rules to protect networks, and those rules consist of two main sections:

The rule header defines the action to take upon any matching traffic, as well as the protocols, network addresses, port numbers, and direction of traffic that the rule should apply to.

The rule body section defines the message associated with a given rule, and most importantly the payload and non-payload criteria that need to be met in order for a rule to match. Although rule options are not required, they are essential for making sure a given rule targets the right traffic.

2. **Service Rules**

service rules let rule writers target a particular service regardless of the IP addresses or ports being used in a given network flow.

3. **File Rules**

allow rule writers to create rules to match a particular file regardless of the protocol, source IPs, destination IPs, ports, and service.

Snort is able to process files that are sent using any of the following application-layer protocols: HTTP, SMTP, POP3, IMAP, SMB, FTP.

4. **File Identification Rules**

File identification rules take advantage of Snort's detection engine to enable file type identification. These rules are basic Snort 3 rules, but instead of alerting on and/or blocking traffic, they identify files based on the contents of that file and then define a file type that can be used in subsequent rules

Q4

The distance keyword is similar to offset but is relative to a preceding content match instead of the start of the payload/buffer. It tells Snort to look skip X number of bytes after the last content match before looking for this one.

The within keyword is similar to depth but is relative to a preceding content match instead of the start of the payload. It tells Snort to look this content match within X number of bytes of the last one.

The replace rule option is used to overwrite prior matching content with the string provided to the option. This option should be used with the rewrite rule action, and it works for raw packets only.

The http_stat_code sticky buffer contains the status code field of an HTTP response status line. This includes values such as 200, 403, and 404.

The metadata option adds additional and arbitrary information to a rule in the form of key-value pairs

Practical Questions

Part 1: drop nmap ping request

Rule:

drop icmp any any -> 192.168.179.101 any (msg:"Ping request blocked"; sid:10000004;)

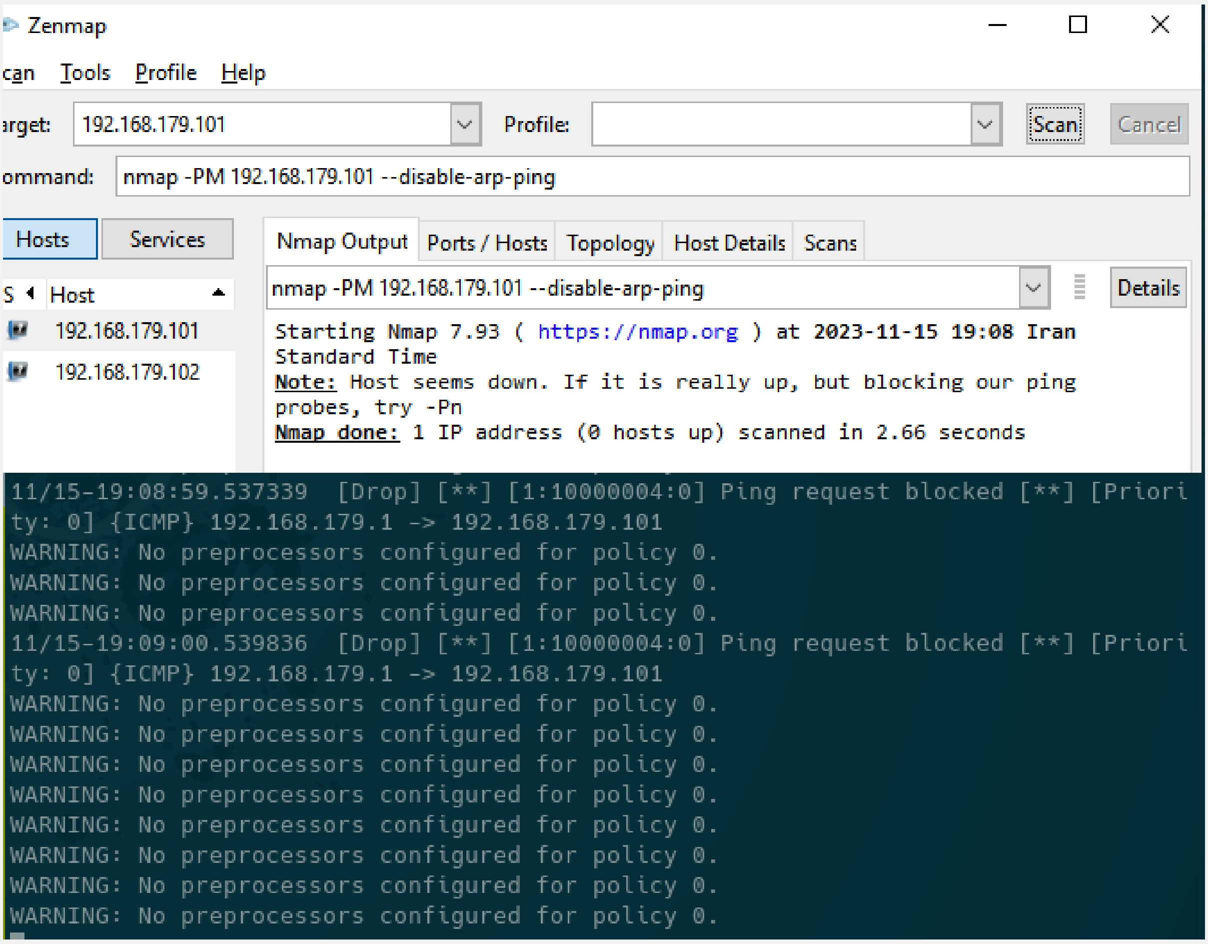
in order to make snort drop the packets, it should be started in inline mode, I used the following flags

snort -Q -A console --daq nfq --daq-var device=enp0s3 --daq-var queue=1

then used these iptables commands to forward packets to nfq

iptables -t nat -I PREROUTING -j NFQUEUE --queue-num 1
iptables -I FORWARD -j NFQUEUE --queue-num 1
iptables -I INPUT -j NFQUEUE --queue-num 1

Result:



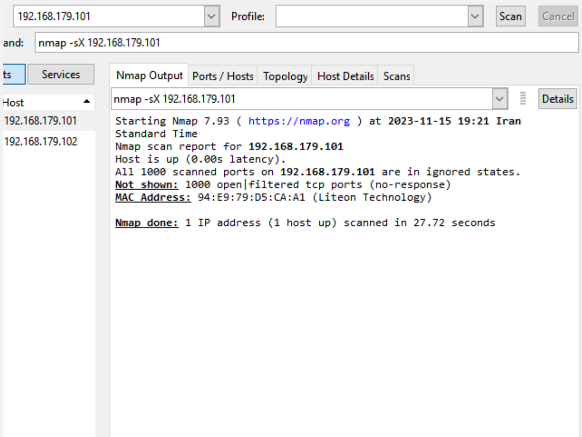
Bypass: we can use -Pn flag in nmap to bypass this rule

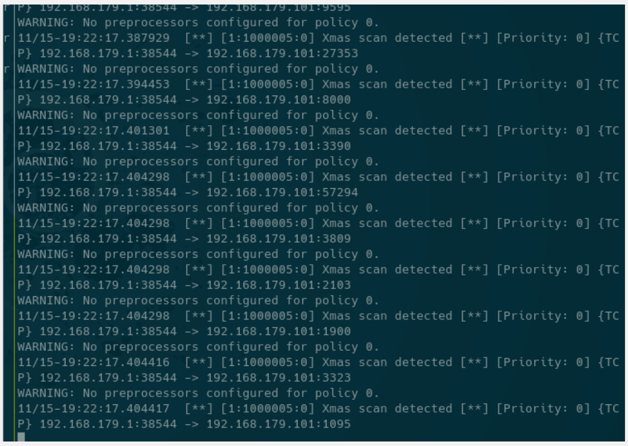
Part 2: Xmas Scan

Rule:

alert tcp any any -> 192.168.179.101 any (flags: FPU; msg:"Xmas scan detected"; sid:1000005;)

Result:



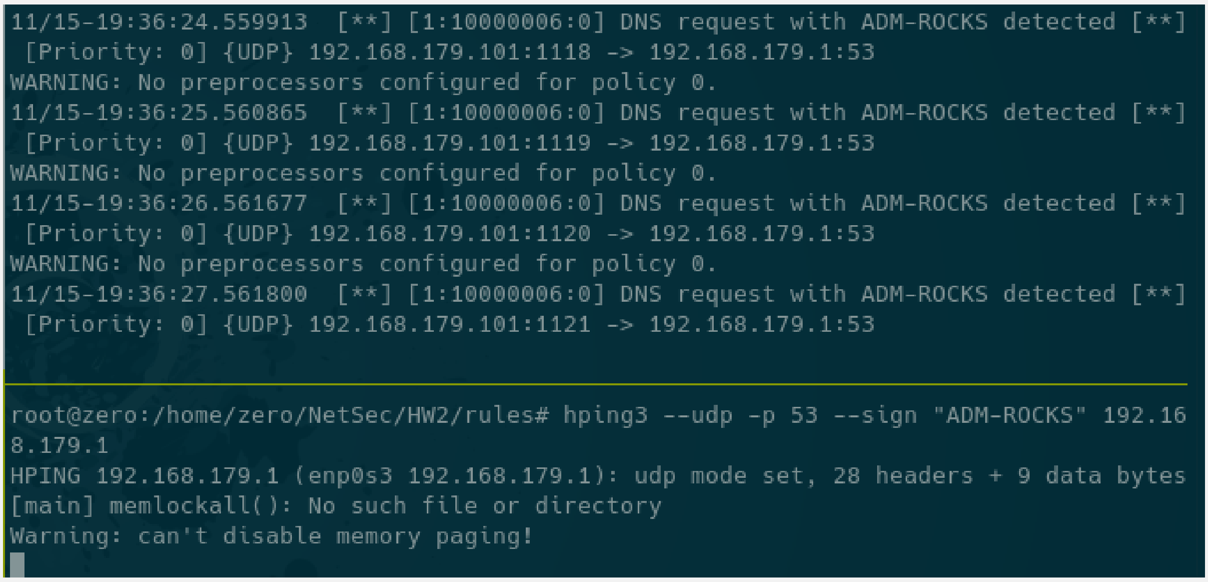


Part 3: ADM-ROCKS DNS Request

Rule:

alert udp 192.168.179.101 any -> any 53 (content:"ADM-ROCKS"; msg:"DNS request with ADM-ROCKS detected"; sid:10000006;)

Result: (using tmux)

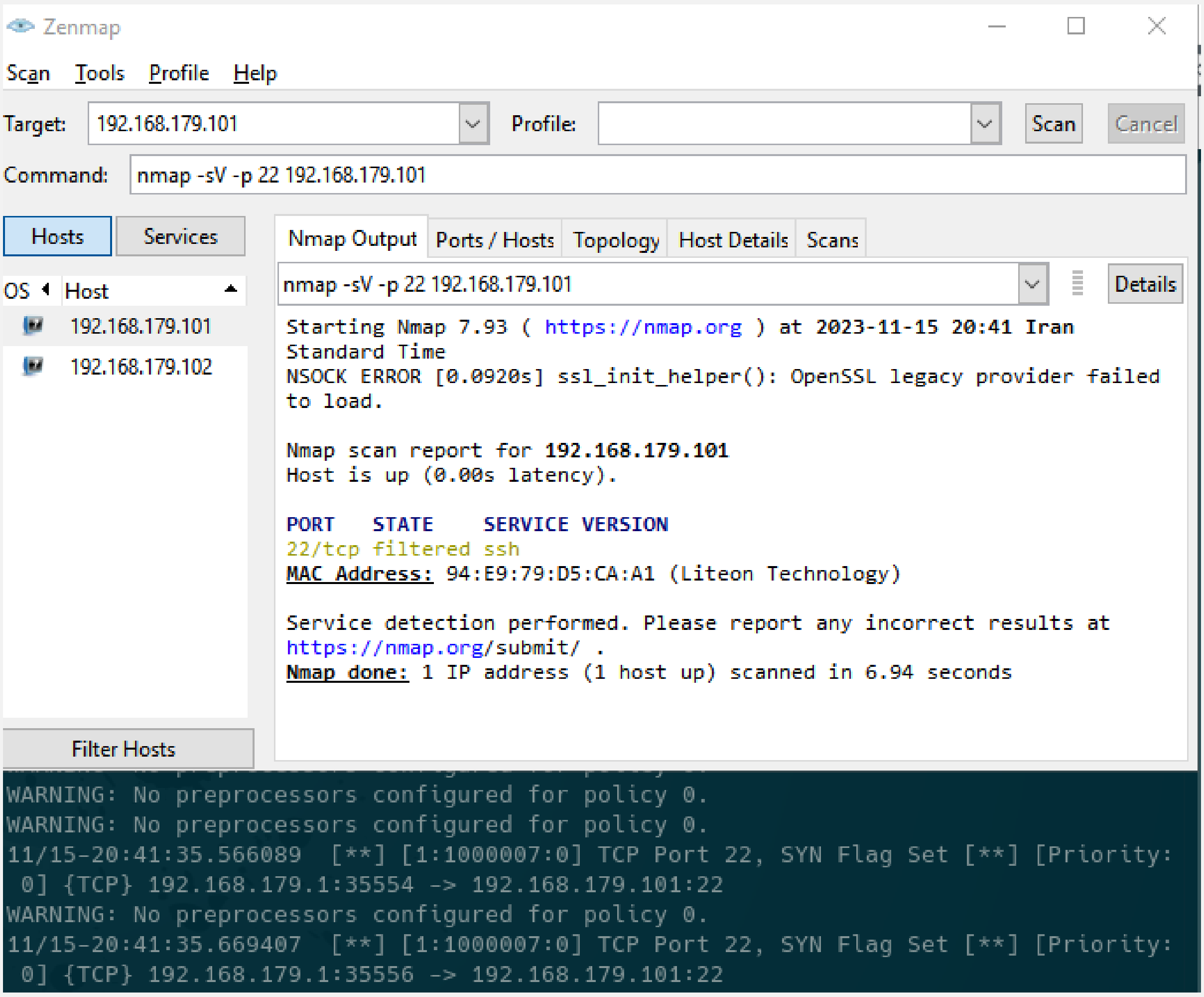


Part 4: nmap SSH scan

Rule:

alert tcp any any -> 192.168.179.101 22 (dsize:0; flags:S; msg:"TCP Port 22, SYN Flag Set, No Payload"; sid:1000007;)

Result:



Part 5: HTTP DoS

Rule:

alert tcp any any -> 192.168.179.101 80 (msg:"Potential HTTP DoS Attack"; threshold:type threshold, track by_src, count 5, seconds 10; flags:!S; sid:100008;)

Test:

hping3 -c 60 -p 80 192.168.179.101

Result:

```
WARNING: No preprocessors configured for policy 0.
11/15-21:11:39.892403  [**] [1:100008:0] Potential HTTP DoS Attack [**] [Priority: 0] {TCP} 192.168.179.104:1895 -> 192.168.179.101:80
WARNING: No preprocessors configured for policy 0.
```

Part 6: SQL Injection

Rule:

alert tcp any any -> 192.168.179.101 any (msg:"SQL Injection Attempt"; flow:established,to_server; content:"' or 1=1--"; sid:1000009;)

Test: (input.txt contains the payload string)

hping3 -c 3 -p 80 -S -d 40 -E input.txt 192.168.179.101

```
root@server1:/home/zero# hping3 -c 3 -p 80 -S -d 40 -E input.txt 192.168.179.101
HPING 192.168.179.101 (enp0s3 192.168.179.101): S set, 40 headers + 40 data bytes
[main] memlockall(): Operation not supported
Warning: can't disable memory paging!

--- 192.168.179.101 hping statistic ---
3 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@server1:/home/zero#
```

Result:

```
11/15-21:47:08.788991  [**] [1:1000009:0] SQL Injection Attempt [**] [Priority: 0] {TCP} 192.168.179.104:2115 -> 192.168.179.101:80
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
WARNING: No preprocessors configured for policy 0.
11/15-21:47:09.789718  [**] [1:1000009:0] SQL Injection Attempt [**] [Priority: 0] {TCP} 192.168.179.104:2116 -> 192.168.179.101:80
WARNING: No preprocessors configured for policy 0.
11/15-21:47:10.790390  [**] [1:1000009:0] SQL Injection Attempt [**] [Priority: 0] {TCP} 192.168.179.104:2117 -> 192.168.179.101:80
```

Part 7: index.php access

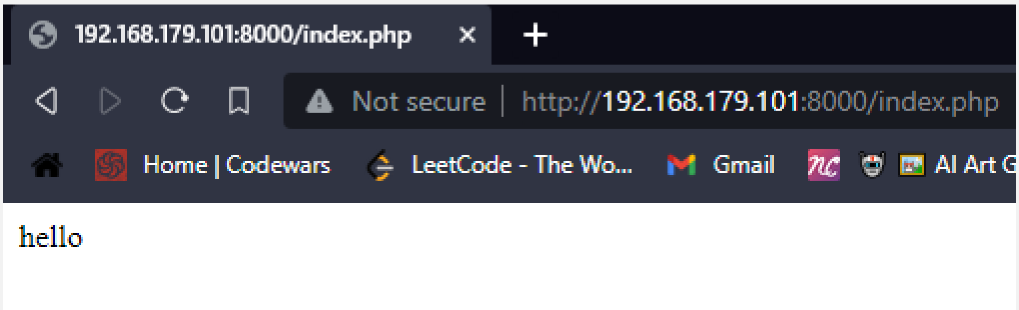
Rule:

alert tcp any any -> 192.168.179.101 8000 (msg:"Unauthorized Access to index.php";flow:to_server,established; content:"GET /index.php"; sid:1000010;)

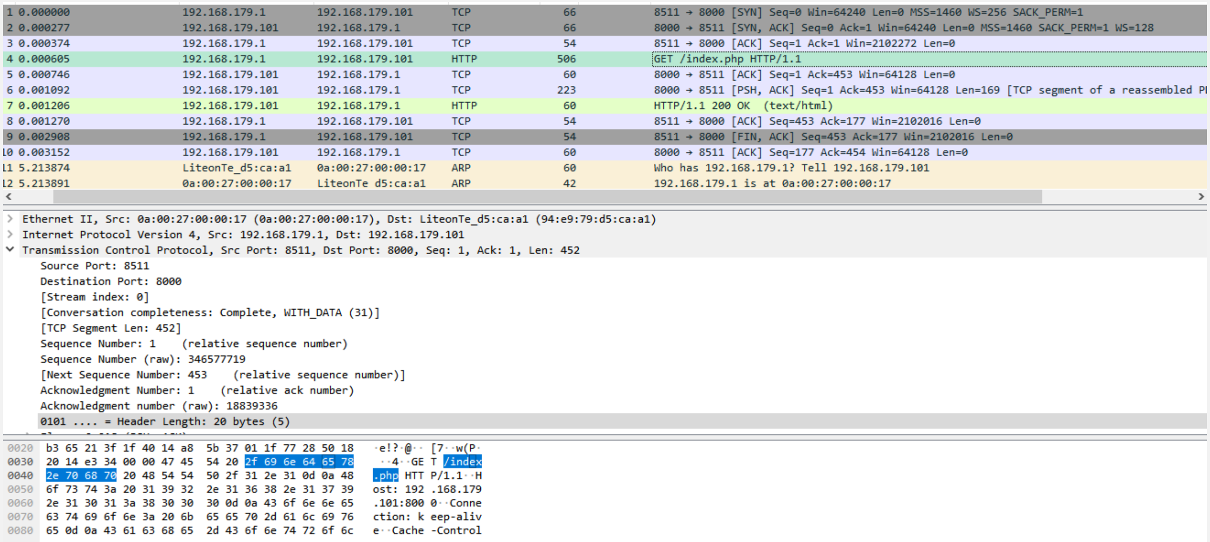
I started a simple php server with a simple index file with this command:

php -S 0.0.0.0:8000

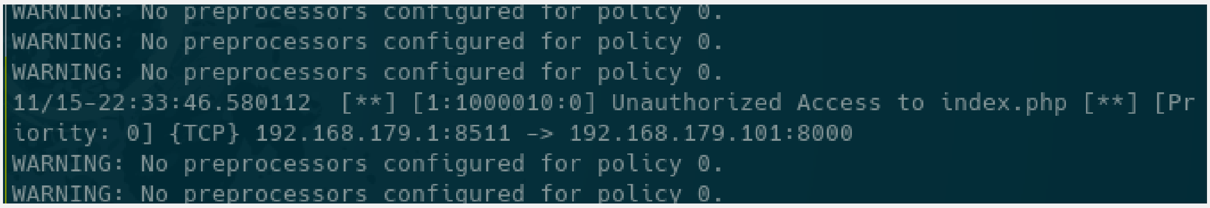
Browser:



Wireshark:



Result:



Bonus 1: Suricata

Suricata is an open-source network threat detection engine developed by the Open Information Security Foundation (OISF). It is designed to monitor network traffic and detect potential security threats or intrusions. Suricata is often used as an intrusion detection system (IDS), intrusion prevention system (IPS), and network security monitoring (NSM) tool.

1. **Multi-Threaded Architecture:**

- Suricata is designed to take advantage of multi-core processors, allowing it to handle high network traffic loads efficiently.

2. **Traffic Capture and Analysis:**

- Suricata can capture and analyze network traffic in real-time using various capture methods, including PCAP files and AF_PACKET.

3. **Rule-Based Detection:**

- Similar to Snort, Suricata uses rule-based detection to identify and alert on malicious activity. It supports the Emerging Threats and VRT rule sets.

4. **Protocol Support:**

- Suricata supports a wide range of network protocols, including IPv4, IPv6, TCP, UDP, ICMP, HTTP, DNS, and more.

5. **File Extraction:**

- Suricata can extract files from network traffic for further analysis, aiding in the identification of malware or other malicious content.

6. **Integration with Other Tools:**

- Suricata can integrate with other security tools and platforms, such as intrusion prevention systems, SIEMs (Security Information and Event Management), and threat intelligence feeds.

7. **Performance:**

- The multi-threaded architecture and efficient handling of network traffic contribute to Suricata's performance, making it suitable for high-speed networks.

Bonus 2: Zeek

Zeek, formerly known as Bro, is an open-source network security monitoring (NSM) framework. Unlike traditional intrusion detection systems (IDS) that focus on signature-based detection (matching known patterns of malicious activity), Zeek is known as an anomaly-based IDS because it primarily relies on the detection of anomalous behavior in network traffic.

The term "anomaly-based IDS" refers to a detection approach that focuses on identifying deviations from normal behavior rather than relying on known attack signatures. In the context of Zeek, the framework is considered anomaly-based for several reasons:

1. **Behavioral Analysis:**

Zeek observes and analyzes the behavior of network traffic, looking for patterns that deviate from the expected or normal baseline. This includes identifying unusual communication patterns, sudden spikes in traffic, or deviations from established network behavior.

2. **Customizable Policies:**

Zeek allows users to define custom policies using its scripting language. This flexibility enables the creation of rules that are specific to the organization's network environment, making it adaptable to unique threats.

3. **Context-Aware Detection:**

By maintaining detailed connection tracking and extracting metadata, Zeek is able to provide context around network activities. This context is crucial for distinguishing between normal and anomalous behavior.

4. **Dynamic Detection:**

Anomaly-based detection is well-suited for identifying emerging threats and zero-day attacks, as it does not rely on pre-existing signatures. Instead, it adapts to changes in network behavior.