Technical University of Crete

School of Electrical and Computer Engineering

Security of Systems and Services - HPY 413

Network traffic monitoring using Snort

Instructor:
Ioannidis Sotiris

Students:
Salom Iochanas
Konstantinos Pisimisis

January 13, 2025

1 Introduction

The objective of this exercise is to familiarise ourselves with the Snort intrusion detection system. In this exercise, we will create custom packages and custom Snort rules, and examine their efficiency. The exercise provides a global overview of how an intrusion detection system would have functioned.

2 Creating Custom Network Traffic

This part of the assigment is solved though python.For completeness, we provide an indicative way on how we construct the requested packets.

Creation of packages

```
def student_packet():
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    payload = f'{STUDENT_NAME} {timestamp}'
    packet = IP(dst="192.168.1.1") / TCP(dport=54321) /
     ↪  Raw(load = payload) #Use RAW since we have a custom
     ↪  payload

    return [packet]
```

3 Creation of custom Snort rules

The rules we created for this part of the exercise are the following.

Custom TCP detector

```
alert tcp any any -> 192.168.1.1 54321 (msg:"Custom
   TCP packet detected"; content:"YourName";
   sid:1000001;)
```

This Snort rule triggers an alert for any TCP packet sent to the destination IP `192.168.1.1` on port `54321` that contains the string `"YourName"` in its payload. The alert message will display as `"Custom TCP packet detected"`, and the rule is identified by SID `1000001`.

Port detection rule

```
alert ip any any -> 192.168.1.2
    [80,443,22,23,21,53,554,1433,3389,1883] (msg:
    "Port can Packet"; sid: 1000002;)
```

This Snort rule triggers an alert for any IP-based protocol(in our case TCP or UDP) packet sent to the destination IP `192.168.1.2` on ports `21,22,23,53,80,443,554,1433,1883,3389`. The alert message will display as `"IP packet to common ports (TCP/UDP)"`, and the rule is identified by SID `1000002`.Last but not least, it is classified as an `attempted-recon` attack type.

Base64 detector

```
alert tcp any any -> 192.168.1.3 8080 (msg: "Base644
    Malicious Packet"; sid: 1000003;
    classtype:misc-activity)
```

This Snort rule triggers an alert for any TCP packet sent to the destination IP `192.168.1.1` on port `8080`. The alert message will display as `"TCP packet detected"`, and the rule is identified by SID `1000003`.Last but not least, it is classified as an `misc-activity` attack type.

DNS detector rule

```
alert udp any any -> 192.168.1.1 53 (msg: "DNS
    Suspicious Domain Packet"; sid: 1000004;
    classtype:attempted-recon)
```

This Snort rule triggers an alert for any UDP packet sent to the destination IP `192.168.1.1` on port `53`. The alert message will display as `"TCP packet detected"`, and the rule is identified by SID `1000004`.Last but not least, it is classified as an `attempted-recon` type.

ICMP detector

```
alert icmp any any -> 192.168.1.4 any (msg: "Ping
   Test Packet"; sid: 1000005;)
```

This Snort rule triggers an alert for any ICMP packet sent to the IP address `192.168.1.4`. The alert message will be ”Ping Test Packet.The rule is identified by SID `1000005`

4 Slammer detection

By analyzing the network traffic captured in the provided pcap file, we observe that the Slammer worm specifically targets port 1434, which is the default port used by the Microsoft SQL Server for the SQL Resolution Service. This is significant because the Slammer worm exploits a vulnerability in the SQL Server's handling of this service, allowing the worm to propagate and initiate denial-of-service attacks.

In order to prevent such attacks, we create the following snort rule:

Slammer detector

```
alert udp any any -> any 1434 (msg:"Slammer Packet";
   sid: 1000006;
   content:"|726e51686f756e746869636b43684765|")
```

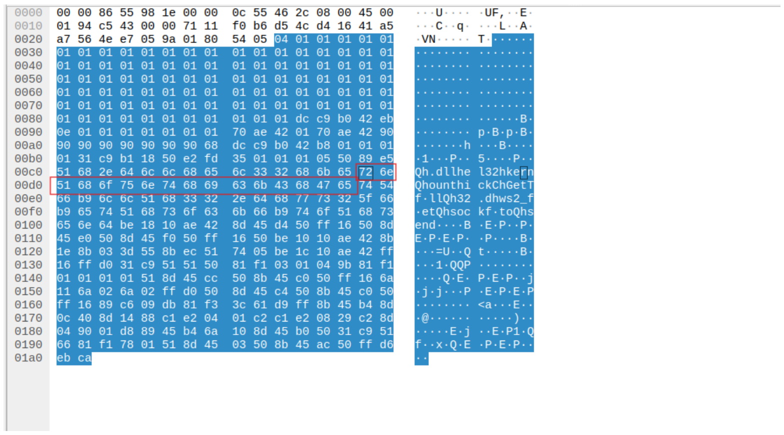

Figure 1: Slammer payload(red box)

This Snort rule triggers an alert for any **UDP** packet sent to **port 1434** that contains the specified hexadecimal value. The alert message will be ”Slammer Packet”, and the rule is identified by SID `1000006`.

The hexadecimal value represents a specific payload associated with the Slammer worm. Omitting this content filter would result in more alerts, potentially triggering false positives, and failing to detect the worm’s specific attack pattern.

5 Evaluation of Snort

In order to evaluate the snort we use a large pcap file(`https://share.netresec.com/s/mQaZcBPAN3iqdYH`)

Using python, we plot the collected data and we receive the following image:

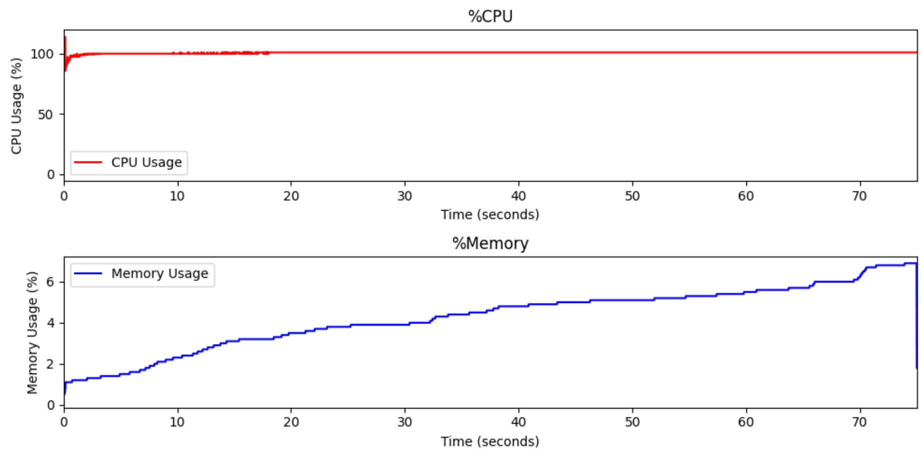

Figure 2: Caption

Evaluation of Snort: As demonstrated by the graphs above, the CPU stabilizes,reaching 100% usage, after a brief period indicating that it is snorting a large file.On the other hand, the memory usage continues to increase until the program reaches its end.

Optimization actions In order to enhance the efficacy of the Snort, it is necessary to refine the existing rules.Initially, the rule with SID: 11000002 can be enhanced using the flag: `flow: to_server` which examines only the connection from server to client.Another useful flag is the `established` which examines only established TCP connections.Though this way we will reduce the times the rule is called.In addition to that, we can use the support for regular expression patterns that is offered by the Snort so we can more easily detect Base64 attacks.

6 Theoritical Questions

6.1 Detection of malicious activity though Snort

Snort uses a rule-based system to detect malicious network activity by analyzing network traffic against predefined rules. Each rule consists of a header (action, protocol, source/destination IPs, and ports) and options (patterns like content matching or flags). When Snort examines packets, it checks if they match any rule conditions. If a match is found, Snort generates an alert or takes an action.

Snort detects threats using signature-based detection (matching known attack patterns), protocol anomaly detection (identifying irregular protocol behavior), and content matching (looking for specific payload patterns). Custom rules can be added, and Snort regularly updates its rule set to stay current with emerging threats.

6.2 Limitations of signature-based intrusion detection systems

When it comes to intrusion detection systems that tend to use signature based models, even though they provide a greate assent, it is necessary to take into account the following limiations:

1. **Depend on known attack signatures:**The signature based model work on already known attacks .Thus, attackers may can often produce variants of known threats which can go undetected.

2. **Requires regular updates to the signature database:** Automation tools have streamlined the process, but network administrators may need to manually update their databases to keep up with the latest threads.

3. **Performance issues:** Due to the increasing size of the signature database, it is possible to create performing issues.

4. **Lack of adaptability:** IDSs that use a signature database cannot adapt to unknown conditions or threads.

5. **False positive rate:** A signature database which is not subject to regular maintenance may result in an elevated rate of false positives, a consequence of overlapping rules.

6. **Limited monitor:** In certain cases, is limited to only header monitoring when dealing with encrypted packets.

6.3 Pros and Cons of using Snort

Using Snort in a real-world scenario offers several advantages and limitations, summarized as follows:

- **Pros**
  - Efficient for real time analysis of a large stream of packets.
  - Quickly identifies known threats, providing a reliable buffer for unsophisticated/copycat attacks.
  - Low false-positive rate due to reliable signature matching
- **Cons**
  - Can be often bypassed by producing malicious packet variants.
  - Needs constant manual updating of the database.
  - Overlapping snort rules can cause issues when database size increases.