SNORT

All Slide Shows: https://lab.dylantran.tech/

Topics

- What is Snort?
- Snort Architecture
- Snort Modes of Operation
- Snort Rule
- Installation

Requirement

- Understanding Snort
- Installation and configuration Snort
- Writing some basic rules
- Writing rules for detecting vul, malicious activity that you analyzed in Lab-work 2

What is Snort?

SNORT

- **Snort** is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks.
- **Snort** can be configured in two main modes: Intrusion Detection System (IDS), which passively alerts administrators about potential security issues, and Intrusion Prevention System (IPS), which actively blocks malicious traffic.

Intro to Snort

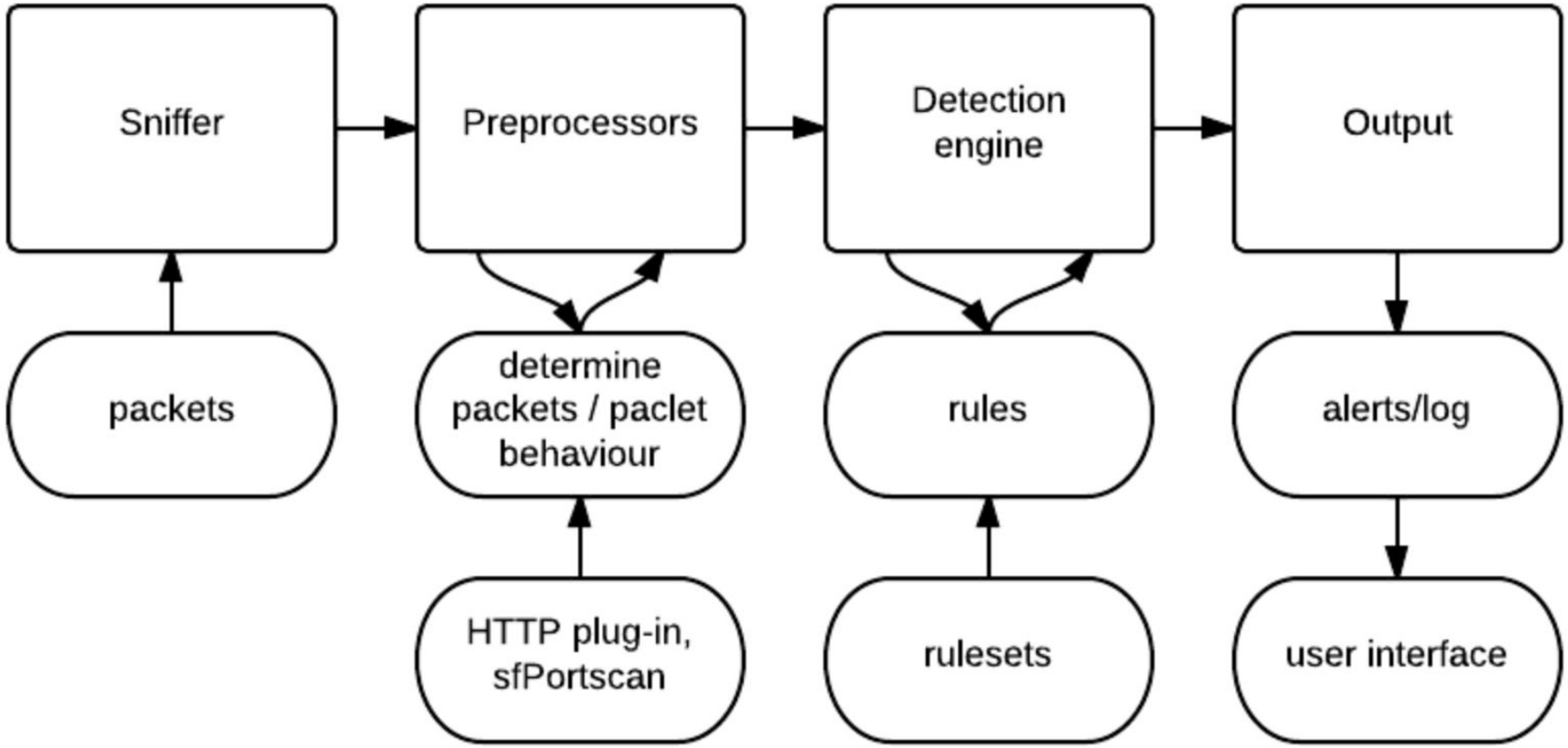
- Snort modes:
  - Snort is a multi-mode packet analysis tool
    - Sniffer
    - Packet Logger
    - Forensic Data Analysis tool
    - Network Intrusion Detection System
- Where did it come from?
  - Developed out of the evolving need to perform network traffic analysis in both real-time and for forensic post processing

Snort “Metrics”

- Small (~800k source download)
- Portable (Linux, Windows, MacOS X, Solaris, BSD, IRIX, Tru64, HP-UX, etc)
- Fast (High probability of detection for a given attack on 100Mbps networks)
- Configurable (Easy rules language, many reporting/logging options
- Free (GPL/Open Source Software)

Snort Architecture

Data Flow


Sniffer
Preprocessors
Detection engine
Output
packets
determine packets / paclet behaviour
rules
alerts/log
HTTP plug-in, sfPortscan
rulesets
user interface

Packet Capture Layer

The Packet Capture Layer is the first step in Snort's architecture, where network traffic is captured from the network interface. It uses the libpcap library to intercept packets and pass them to the subsequent layers for analysis.

- Interfaces with network to capture packets
- Uses libpcap/WinPcap for raw packet access

Preprocessors

Preprocessors are modules that analyze and manipulate network traffic before it reaches the detection engine. They prepare the data for more accurate and efficient intrusion detection by reassembling packets, decoding protocols, and identifying patterns.

- Packet Normalization: Adjusts packet data to ensure consistency and correct interpretation.
- Protocol Decoding: Decodes specific protocols (HTTP, FTP, DNS, etc.) to make them easier for Snort to analyze.
- Stream Reassembly: Rebuilds packet streams (such as TCP or UDP) to detect attacks that span multiple packets.
- Traffic Analysis: Identifies suspicious or malicious traffic patterns.
- Protocol Anomaly Detection: Identifies unusual or malformed protocol behavior.

Detection Engine

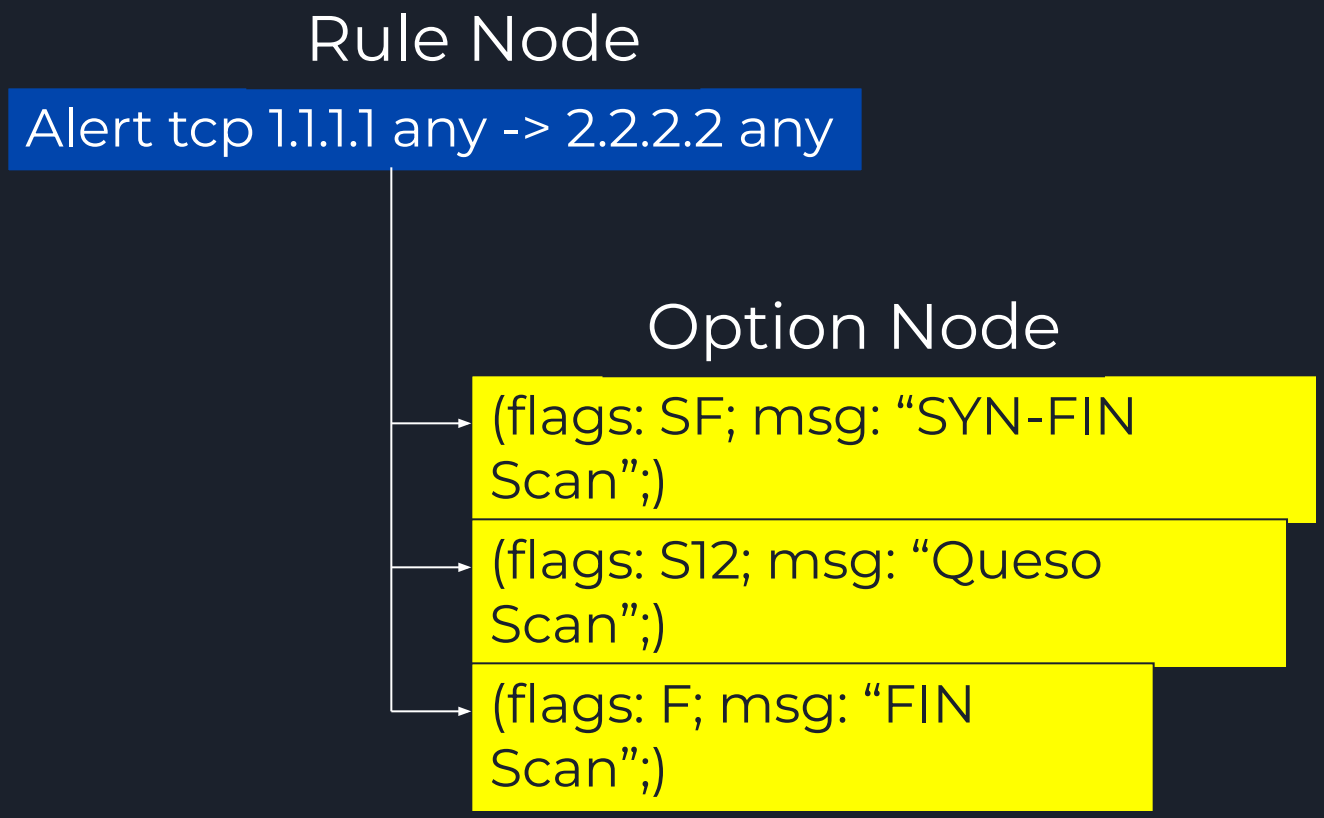
The Detection Engine is the core of Snort's intrusion detection capabilities. It analyzes the processed network traffic (from the Packet Capture and Preprocessor layers) to detect signs of malicious activity based on predefined rules.

- Signature-Based Detection: Matches network traffic against predefined attack signatures.
- Anomaly-Based Detection: Identifies traffic patterns that deviate from established baselines (normal behavior).
- Stateful Inspection: Tracks the state of connections and analyzes traffic within the context of the entire session.
- Rule Processing: The engine processes Snort rules to determine if a packet matches a threat pattern.

Detection Engine: Rules

Rule Header	Rule Options
Alert tcp 1.1.1.1 any -> 2.2.2.2 any	(flags: SF; msg: “SYN-FIN
Alert tcp 1.1.1.1 any -> 2.2.2.2 any	(flags: S12; msg: “Queso
Alert tcp 1.1.1.1 any -> 2.2.2.2 any	(flags: F; msg: “FIN Scan”;)

Detection Engine: Internal Representation



Output Modules

Output modules in Snort are responsible for logging detected events and generating alerts. They define how and where Snort's results are saved or sent, allowing integration with other systems for analysis, monitoring, and response.

- Alert Output: Generates alerts when Snort detects a threat.
- Log Output: Logs packet data or summary information for future analysis.
- Database Output: Stores event data in relational databases for long-term analysis and reporting.
- Third-Party Integration: Integrates with external systems for further analysis and action.

Snort Modes of Operation

Snort Modes

- Three main operational modes
  - Sniffer Mode
  - Packet Logger Mode
  - NIDS Mode
  - Inline Mode (IPS)
- Operational modes are configured via command line switches
  - Snort automatically tries to go into NIDS mode if no command line switches are given, looks for snort.conf configuration file in /etc/snort

Sniffer Mode

Snort's Sniffer Mode is a passive mode where Snort captures and inspects network traffic without performing any analysis or detection. It is useful for monitoring network activity in real time, but it does not generate alerts or logs based on the traffic.

- Real-Time Packet Capture: Captures network packets as they traverse the network.
- Packet Display: Shows raw packet data on the screen for immediate visibility.
- No Analysis or Filtering: Only captures packets, without applying any detection rules or analysis.
- Network Monitoring: Useful for basic traffic monitoring and troubleshooting.
- Lightweight Operation: Minimal processing overhead, as no detection engine is used.

Packet Logger Mode

Packet Logger Mode in Snort allows Snort to capture and log network packets to disk for later analysis. It is a passive mode where Snort captures raw packets without performing detection or generating alerts, but logs the traffic for future review.

- Stores Packets to Disk: Saves network packets to files for later analysis.
- Detailed Traffic Records: Logs complete packet data, including headers and payload.
- Supports Forensic Analysis: Enables in-depth examination of past network activity.
- Configurable Storage: Allows specifying log formats and storage locations.
- Useful for Audit Trails: Creates a reliable record of network events for compliance or investigation.

NIDS Mode

NIDS (Network Intrusion Detection System) Mode is the standard operating mode of Snort for real-time intrusion detection. In this mode, Snort actively analyzes network traffic and generates alerts for suspicious or malicious activities based on predefined rules.

- Real-Time Threat Detection: Monitors live traffic for signs of attacks.
- Rule-Based Analysis: Uses signature-based and anomaly-based rules to detect suspicious activity.
- Alert Generation: Triggers alerts on detecting potential threats or intrusions.
- Network Visibility: Provides insights into network behavior and security posture.
- No Traffic Blocking: Passive detection only—alerts without blocking or modifying traffic.

Inline Mode (IPS)

In Inline Mode, Snort acts as an Intrusion Prevention System (IPS), actively blocking or modifying traffic in real-time to prevent malicious activity from reaching its target. Unlike in NIDS mode, where Snort only detects threats, Inline Mode allows Snort to take action, such as blocking or dropping harmful packets.

- Active Traffic Blocking: Detects and blocks malicious traffic in real-time.
- Intrusion Prevention: Acts as a firewall, preventing attacks from reaching targets.
- Customizable Rule Actions: Rules can drop, reject, or log traffic based on threat level.
- Packet Modification: Modifies or removes harmful data before it reaches endpoints.
- Enhanced Security Control: Proactively protects network by stopping threats instantly.

Snort Rules

Snort Rules

- Snort rules are extremely flexible and are easy to modify, unlike many commercial NIDS
- Sample rule to detect SubSeven trojan:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP vsFTPd 2.3.4
    backdoor login attempt"; flow:to_server,established; content:":)";
    content:"USER "; offset:0; depth:5; nocase; classtype:attempted-admin;
    sid:1000002; rev:1;)
```

- Elements before parentheses comprise ‘rule header’
- Elements in parentheses are ‘rule options’

Snort Rules

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP vsFTPd 2.3.4 backdoor login attempt"; flow:to_server,established; content:":)"; content:"USER "; offset:0; depth:5; nocase; classtype:attempted-admin; sid:1000002; rev:1;)

- alert tcp $EXTERNAL_NET any -> $HOME_NET 21: Monitors any TCP traffic from external sources to your home network on FTP port 21.
- msg:"FTP vsFTPd 2.3.4 backdoor login attempt": Provides a message for the alert when this rule is triggered.
- flow:to_server,established: Ensures that the rule applies only to traffic going to the server as part of an established connection.
- content:":)";: Matches the :) smiley face, which is the trigger for the vsFTPd backdoor.
- content:"USER "; offset:0; depth:5; nocase;: Checks that the smiley face is part of the USER command, which is standard in FTP login attempts.
- classtype:attempted-admin: Classifies this as an administrative attempt (backdoor access).
- sid:1000002; rev:1;: Assigns a unique Snort ID for the rule and sets the revision number.

Snort Rules

- bad-traffic.rules
- finger.rules
- smtp.rules
- dos.rules
- tftp.rules
- web-frontpage.rules
- web-attacks.rules
- icmp.rules
- backdoor.rules
- info.rules
- virus.rules

exploit.rules
ftp.rules
rpc.rules
ddos.rules
web-cgi.rules
web-iis.rules
sql.rules
netbios.rules
shellcode.rules
icmp-info.rules
local.rules

scan.rules
telnet.rules
rservices.rules
dns.rules
web-coldfusion.rules
web-misc.rules
x11.rules
misc.rules
policy.rules

attack-responses.rules

See it in /etc/snort/rules

Installation

Installation

Run with root privileges (sudo)

- Install using packages: $ apt install snort -y (the easy way but not using in this lab)
- Build and install from source code: (using the way)

Step 1: Update and install support packages: Pcap, PCRE, libdnet, DAQ, lib.

```
$ apt install -y build-essential
$ apt install -y libpcap-dev libpcre3-dev libdumbnet-dev
$ apt install -y bison flex libluajit-5.1-2-dev
$ apt install -y zlib1g-dev liblzma-dev openssl libssl-dev libnghttp2-dev
```

Step 2: Create a directory containing the installation source code and install.

```
$ mkdir -p ~/snort_src
$ cd ~/snort_src
```

Step 3: Download, unzip, install DAQ package:

```
$ wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz
$ tar -xzvf daq-2.0.7.tar.gz
$ cd daq-2.0.7
$ ./configure && make && make install
```

Installation

Run with root privileges (sudo)

- Build and install from source code:

Step 4: Download Snort.

```
$ wget https://www.snort.org/downloads/snort/snort-2.9.20.tar.gz
$ tar -xzvf snort-2.9.20.tar.gz
$ cd snort-2.9.20
$ ./configure --enable-sourcefire && make && make install
$ ldconfig # update library.
```

Step 5: Create Snort directories and directories containing rule sets.

```
$ mkdir /etc/snort
$ mkdir /etc/snort/rules
$ mkdir /etc/snort/rules/iplists
$ mkdir /etc/snort/preproc_rules
$ mkdir /usr/local/lib/snort_dynamicrules
$ mkdir /etc/snort/so_rules
```

Installation

Run with root privileges (sudo)

- Build and install from source code:

Step 6: Create a file to store rules and IP lists.

```
$ touch /etc/snort/rules/iplists/black_list.rules
$ touch /etc/snort/rules/iplists/white_list.rules
$ touch /etc/snort/rules/local.rules
$ touch /etc/snort/sid-msg.map
```

Step 7: Create log storage directory.

```
$ mkdir /var/log/snort
$ mkdir /var/log/snort/archived_logs
```

Step 8: Setting up username and setting the permissions

```
$ useradd snort
$ chmod -R 5775 /etc/snort
$ chmod -R 5775 /var/log/snort
$ chmod -R 5775 /var/log/snort/archived_logs
$ chmod -R 5775 /etc/snort/so_rules
$ chmod -R 5775 /usr/local/lib/snort_dynamicrules
```

Installation

Run with root privileges (sudo)

- Build and install from source code:

Step 8: Setting up username and setting the permissions

```
$ chown -R snort:snort /etc/snort
$ chown -R snort:snort /var/log/snort
$ chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Step 9: Copy the configuration files from the download folder.

```
$ cd ~/snort_src/snort-2.9.20/etc/
$ cp *.conf* /etc/snort
$ cp *.map /etc/snort
$ cp *.dtd /etc/snort
$ cd
../src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
$ cp * /usr/local/lib/snort_dynamicpreprocessor
```

Installation

Run with root privileges (sudo)

- Build and install from source code:

Step 10: Edit configurations in snort.conf.

$ vim /etc/snort/snort.conf

edit HOME_NET:

```
# Setup the network addresses you are protecting
ipvar HOME_NET [10.10.1.1/32,172.16.1.0/24]
```

Installation

Run with root privileges (sudo)

- Build and install from source code:

Last step: Verify and test configuration.

```
$ snort -T -c snort.conf
```

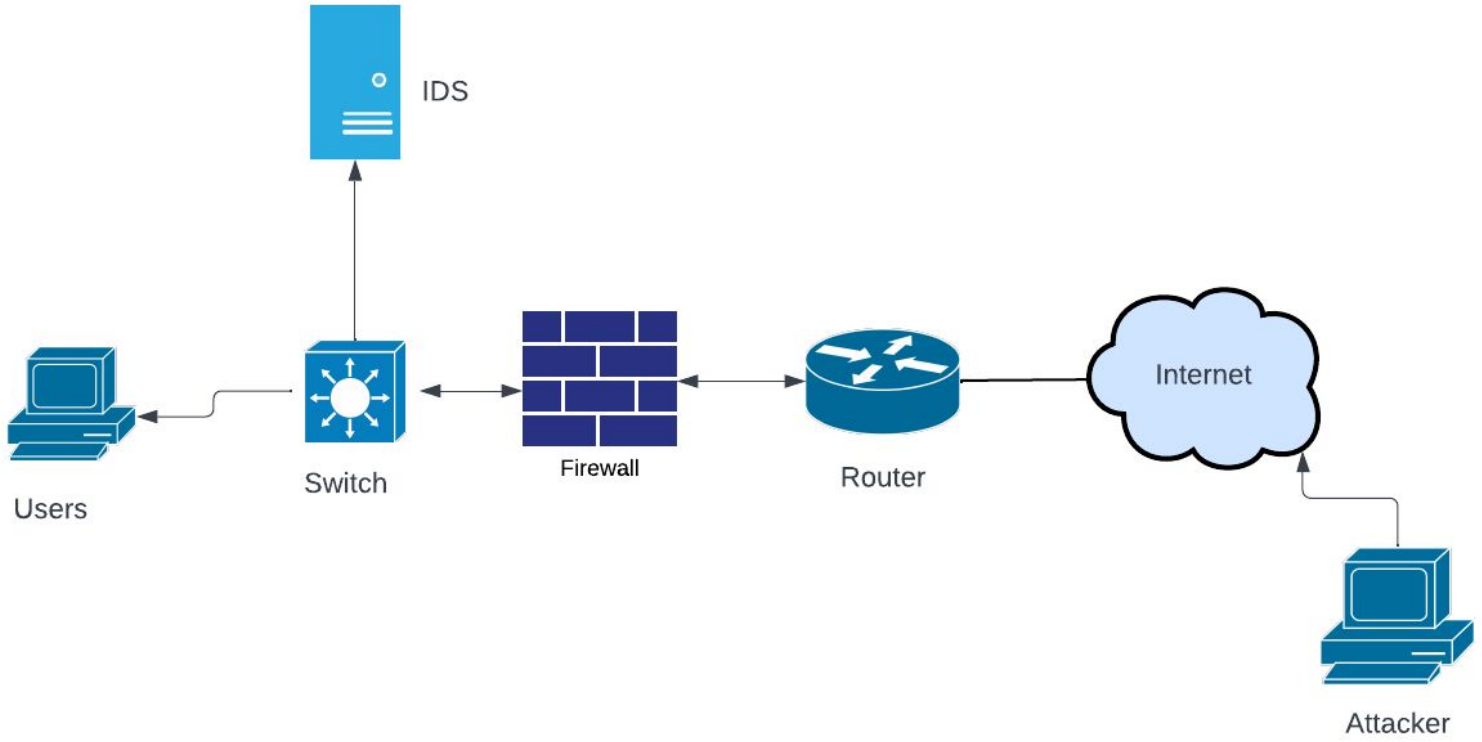
```
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
.............
Total snort Fixed Memory Cost - MaxRss:61384
Snort successfully validated the configuration!
Snort exiting
```

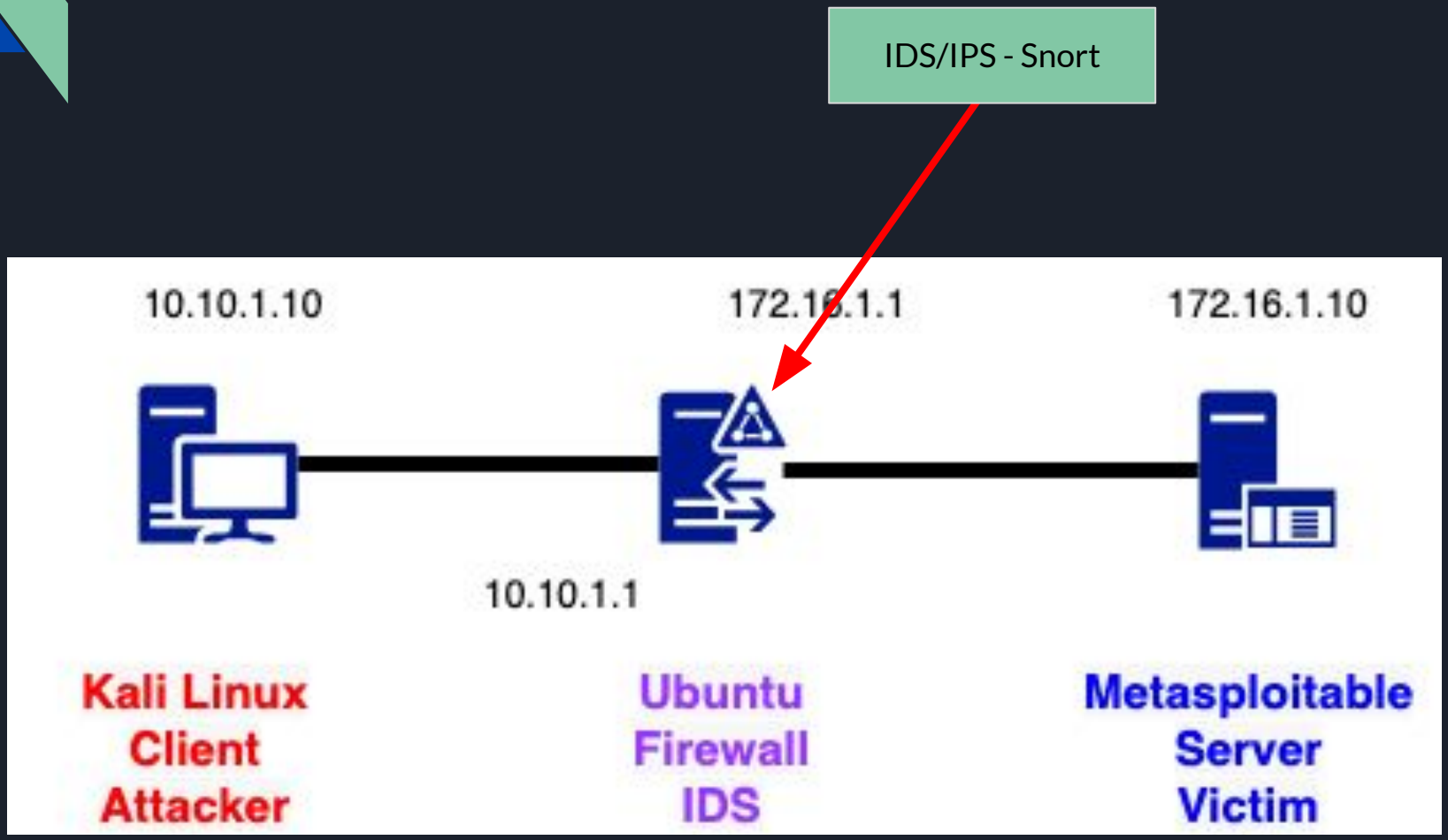
Conclusion

- Snort is a powerful tool, but maximizing its usefulness requires a trained operator
- Becoming proficient with network intrusion detection takes 12 months; “expert” 24-36?
- Snort is considered a superior NIDS when compared to most commercial systems
- Managed network security providers should collect enough information to make decisions without calling clients to ask what happened

How we use the Snort?



How we use the Snort?



References

https://www.snort.org/

https://upcloud.com/resources/tutorials/install-snort-ubuntu

Thank you for listening