Intrusion Detection with Snort

Learning outcome checklist

- LO1: Know how packets flow in Snort
- LO2: Know what can be configured in Snort and how
- LO3: Configure the outputs and know where to view outputs
- LO4: Run Snort in different modes
- LO5: Write simple Snort rules

Reading

Snort user manual: read

- section 1.1, 1.2, 1.3, 1.4, 1.6 (for LO4)
- Section 2.6.1, 2.6.2, 2.6.3 (for LO3)
- Section 3.1, 3.2, 3.3, 3.4.1, 3.4.4, 3.4.5, 3.4.7, 3.5.1, 3.5.5, 3.5.6, 3.5.7, 3.5.8, 3.5.9, 3.5.10, 3.5.23, 3.6.8 (for LO5)

Overview

- Introduction
- Writing Snort rules
- Configuration, deployment

Introduction

SNORT – MODES – PLUGINS - FLOW

Snort



- Free (GPL) misuse-based network IDS/IPS
  - Released in 1998 by Martin Roesch
  - Current versions available for many platforms
    - E.g. Linux, Windows, MacOS X, BSD
  - Official snort ruleset available for subscription and registration users
    - Additional rulesets available
    - Documentation available in many languages (e.g. Japanese)

Snort Intrusion detection

- Freeware Snort
  - Host based
  - Network based
  - GUI interface
  - Rules based
    - can be added, forum
  - Centralized DB plug-ins for MYSQL, Oracle
  - Public support
- Commercial IDS
  - Host based
  - Network based
  - GUI interface
  - User defined
  - Express Updates
  - Centralized DB
  - Rules received 30 days before freeware version

Ruleset categories

Run the provided Virtual machine

Open the file /etc/snort/snort.conf

Menti

Go to https://www.snort.org/rules_explanation to find out the details about each ruleset category

Ruleset categories

Attack-Response Rules: These are designed to catch the results of a successful attack. Things like "id=root", or error messages that indicate a compromise may have happened. Note: Trojan and virus post-infection activity is included generally in the VIRUS ruleset, not here.

BotCC Rules: These are auto-generated from several sources of known and confirmed active Botnet and other Command and Control hosts. Updated daily, primary data source is Shadowserver.org.

Compromised Rules: This is a list of known compromised hosts, confirmed and updated daily as well. This set varied from a hundred to several hundred rules depending on the data sources. This is a compilation of several private but highly reliable data sources. Warming: Snort does not handle IP matches well load-wise. If your sensor is already pushed to the limits this set will add significant load. We recommend staying with just the BotCC rules in a high load case.

Current_Events Rules: These are not long term, or need to be tested before they are considered for inclusion. Most often these will be simple sigs for the Storm binary URL of the day, sigs to catch CLSID's of newly found vulnerable apps where we don't have any detail on the exploit, etc.

Ruleset categories

DOS Rules: Intended to catch inbound DOS activity, and outbound indications. Relatively self-explanatory.

DROP Rules: This is a daily updated list of the Spamhaus DROP (Don't Route or Peer) list. Primarily known professional spammers. More info at http://www.spamhaus.org

DShield Rules: Daily updated list of the DShield top attackers list. Also very reliable. More indo at http://www.dshield.org

Exploit Rules: Rules to detect direct exploits. Generally if you're looking for a windows exploit, Veritas, etc, they'll be here. Things like SQL injection and the like, whie they are exploits, have their own category.

Game Rules: World of Warcraft, Starcraft, and other popular online games have sigs here. We don't intend to label these things evil, just that they're not appropriate for all environments.

Inappropriate Rules: Porn, Kiddy porn, sites you shouldn't visit at work, etc. Warning: These are generally quite Regex heavy and thus high load and frequent false positives. Only run these if you're really interested.

Ruleset categories

Malware Rules:. This set was originally intended to be just spyware, but now covers everything

P2P? Rules: Peer to Peer Bittorrent, Gnutella, Limewire, … not appropriate some networks

Policy Rules: Rules for company or organizational policy. (Myspace, Ebay…)

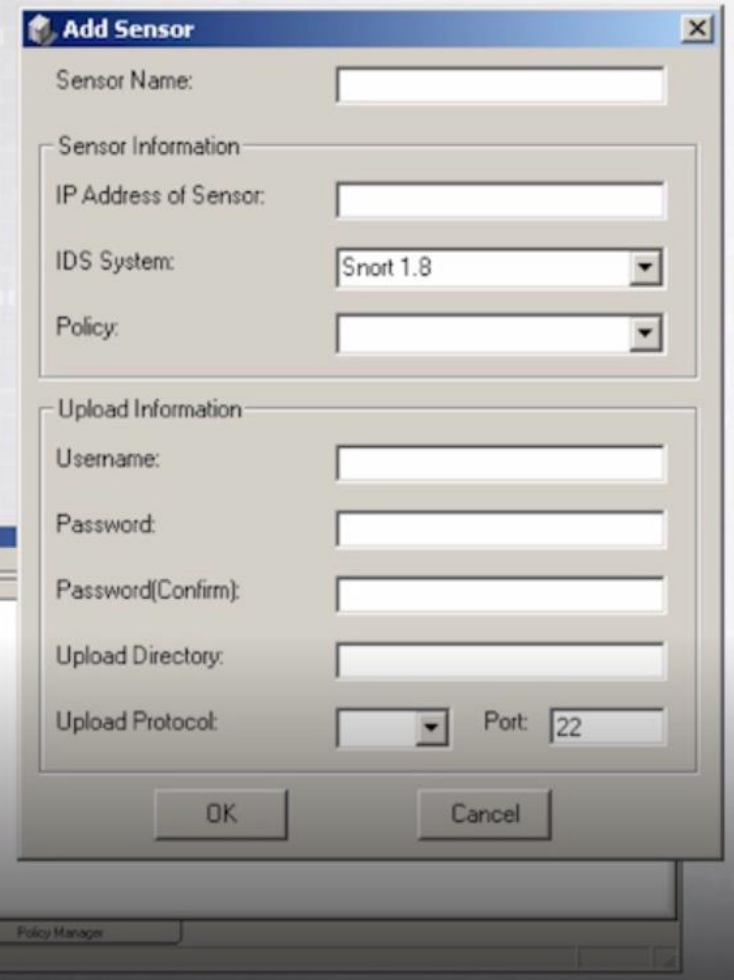
Scan Rules: detect reconnaissance/probing. Nessus, Nikto, portscanning... Early warning

VOIP Rules: A new/emerging ruleset. Small at the moment

Web Rules: Some SQL Injection, web server overflows, vulnerable web apps, that kind of thing. Very important if you're running web servers, and pretty reasonable load.

Web-SQL-Injection Rules: This is a large ruleset that intends to catch specific attacks on specific applications. There are some general SQL injection rules that work pretty well to catch most of what's covered here. But these rules are much more specific to apps and web servers. Run this if you run a highly critical web farm, or are interested in having exact information about incoming web attacks.

Snort Policy Management



- Easy to use Policy Editor
- No version control
- GUI similar for all devices
- Express updates easy
- Large deployments may wish to use CLI

Snort Response

Depends on setup and equipment

- Snort IDS can be set up to become IPS
- Observe to build new rules
- Data collection (additional) if something suspicions
- Automated active response tied to alert

Snort modes of operation

- Modes of operation determined by command line switches
  - Sniffer mode: Sniff and dump packets to screen (e.g. snort –dve)
  - Packet logger mode: Output packets to a log file (e.g. snort –l /usr/var/log)
    - Binary logging by default (pcap format)
    - ASCII logging (-K ascii)
  - NIDS mode: runs snort with ruleset, plug-ins, to monitor traffic for suspicious activity (e.g. snort –c snort.conf)

Exercise

Open the virtual machine

Run

1. sudo snort -vde
2. mkdir log

sudo snort -de –l log

3. Go to the folder Desktop/data/2-running-snort/run-snort,

snort –c snort.conf

Sniffer mode parameters:

-v -Verbose. Display the TCP/IP output in the console.

-d -Display the packet data (payload).

-e -Display the link-layer (TCP/IP/UDP/ICMP) headers.

-**X -**Display the full packet details in HEX.

-**i -**This parameter helps to define a specific network interface to listen/sniff. Once you have multiple interfaces, you can choose a specific interface to sniff.

Packet logger parameters:

-l -Logger mode, target log and alert output directory. Default output folder is **/var/log/snort.** The default action is to dump as tcpdump format in **/var/log/snort**

-K ASCII- Log packets in ASCII format.

-r -Reading option, read the dumped logs in Snort.

-n -Specify the number of packets that will process/read. Snort will stop after reading the specified number of packets.

-c Defining the configuration file.

-T Testing the configuration file.

-N Disable logging.

-D Background mode.

-A Alert modes;

- **Full:** Full alert mode. This mode is also the default; once you use -A and don't specify a mode, Snort defaults to this mode.
- **Fast: fast mode displays the alert message, timestamp, source and destination IP addresses**, along with port numbers.
- **Console:** Provides fast style alerts on the console screen.

-r / --pcap-single= Read a single pcap
--pcap-list="” Read pcaps provided in command (space separated).
--pcap-show Show pcap name on console during processing.

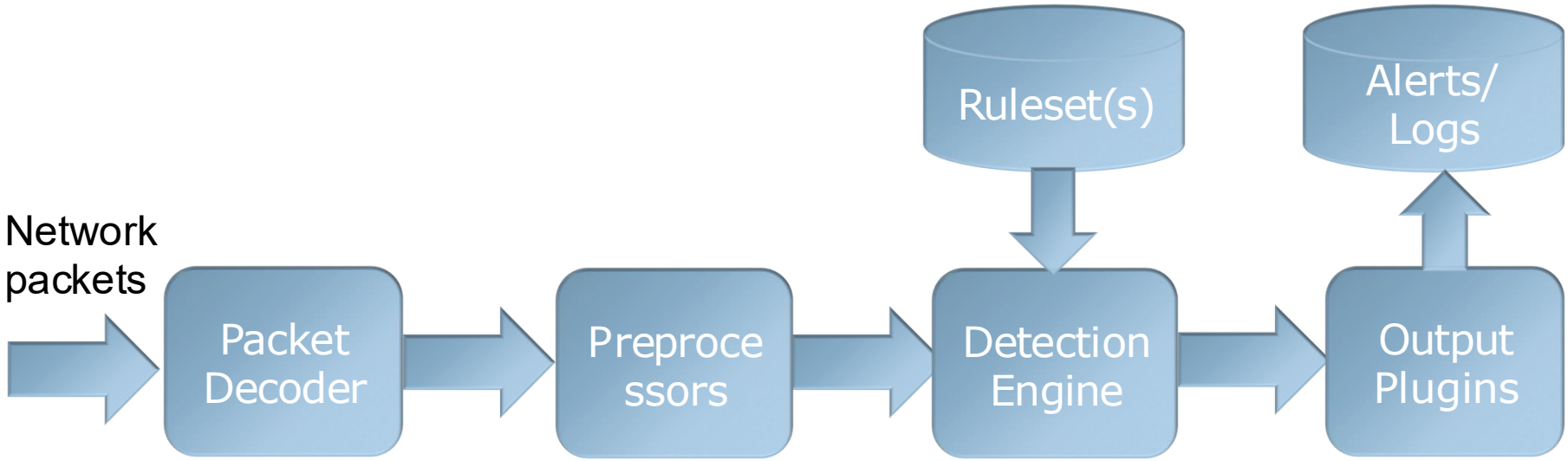
Snort Plug-ins

- Plug-ins are modular pieces of code that extend the basic functionality of snort
  - Preprocessors
    - Examine packets before they are passed to the detection engine
  - Detection
    - Perform tests on a single packet, based on rules
  - Output
    - Report results from preprocessors and detection engine

Exercise

Menti

Packet Flow in Snort



Preprocessors

- Normalize traffic, ensuring that the data in the packets Snort is watching will have the best chance of being in a format that the signatures will recognize
- Self-defense
- Extend Snort's ability to detect network anomalies that may be signs of intrusion—not just notice things that are contained in the rule sets

Preprocessors

- Snort 2.8 supports the following preprocessors:
  - **http_inspect**: decodes web traffic for signature analysis
  - **SFportscan**: portscan detector
  - **Frag3**: performs IP defragmentation
  - **Stream4** and **Stream5**: TCP & UDP stream reassembly
  - **Bo**: Back Orifice detection
  - **Telnet_decode**: telnet negotiation code normalisation
  - **RPC_decode**: RPC normalisation
  - **Statistical analysis**: conversation, performance monitor
  - Dynamic preprocessors
    - FTP_TELNET, SSH, SMTP, DCERPC, DNS

Example

Scenario: a Snort rule is looking for the content /users.pwd in a packet

Attack: An attacker fragments a packet with the content /users.pwd into 2 packets /user and s.pwd

Frag3 preprocessor: reassemble the fragments into a whole and can see the string /users.pwd, avoiding the attack

Output Plugins

- Alert_fast: fast text file alerting (-A fast)
- Alert_full: text alerting with full headers (-A full)
- Alert_smb: SMB Winpopup alerting (-M)
- Alert_syslog: alert to syslog (-s)
- Alert_prelude: alert to the PRELUDE IDS system
- Alert_Unixsock: alert to Unix socket (-A unixsock)
- Log_tcpdump: log packets to binary tcpdump file (-l)
- Database: write alerts or logs to a database
- XML: write alerts or logs to an XML file
- SNMP: generate SNMP traps
- Unified: Use unified binary logging format

Binary and ASCII logging

Binary logging

- Binary log files are in pcap format
  - Supported by hundreds of tools, such as wireshark, tcpdump, etherape, netdude, tcpreplay, chaosreader, etc
- Can be read back from snort using the –r command switch

ASCII logging

- Creates a file per protocol/ port pair
  - PROTOCOL: sourceport-destinationport
  - E.g. tcp:1029-80
- What happens in a portscan?
  - Better to use binary logging for long term applications

Alert modes

- Fast alert mode. Writes the alert in a simple format with a timestamp, alert message, source and destination IPs/ports.
  - Command line: -A fast
  - Preprocessor example: output alert_fast: alert.fast
- Full alert mode. This is the default alert mode and will be used automatically if you do not specify a mode.
  - Command line: -A full
  - Preprocessor example: output alert_full: alert.full
- Syslog mode. Send alerts to syslog (standard for message logging) (e.g. either /var/log/auth.log or /var/log/syslog depending on the system configuration)
  - Command line: -s
  - Preprocessor example: output alert_syslog: LOG_AUTH LOG_ALERT

Alert Modes example

- Full

[**] [1:1149:2] WEB-MISC count.cgi access [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/23-00:50:40.355272 10.1.1.2:1971 -> 10.1.1.4:80
TCP TTL:64 TOS:0x0 ID:25791 IpLen:20 DgmLen:163 DF
AP Seq: 0xD7A171BB Ack: 0x67235EB9 Win:
0x4470 TcpLen: 20
[Xref => http://www.securotyfocus.com/bid/128]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-1999-0021]

- Fast

01/23-00:50:40.355272 [**] [1:1149:2] WEB-MISC count.cgi
access
[**] [Classification: Attempted Information Leak] [Priority: 2]
{TCP} 10.1.1.2:1971 -> 10.1.1.4:80

Exercise

1. Open file local.rules: sudo gedit /etc/snort/rules/local.rules

Add this rule into the file

alert icmp any any -> any any (msg: “ping detected”; sid: 1000001;)

Save and close the file

Run : sudo snort -c /etc/snort/snort.conf -A fast -l log -K ascii

2. Open another Vmware station, open a windows machine

Run: ping to the target machine

3. Go back to the target machine

Shared Object Rules (so_rules)

- Precompiled binary rules
  - Available since Snort 2.6
  - Obfuscate the source code for the rule
    - To provide coverage for zero-day exploits
    - To protect confidential intellectual property with 3rd parties

Snort configuration options

- 'snort.conf' file is the default IDS configuration file
- You can test your configuration/rules file with –T command switch
  - E.g. snort –Tc snort.conf
- Default configuration
  - Log in binary mode.
  - Log directory: '/var/log/snort'
  - Alert mode: Full
  - 'snort.conf' location is in the'/etc' directory

Snort.conf

- Snort.conf allows you to
  - set variables for your network (e.g. home network IP range)
    - var HOME_NET 10.1.1.0/24
  - Configure and tune preprocessors
  - Configure output plugins
  - Customise your ruleset
- Configuration options at 'snort.conf' can be overridden at command line

Conf segments

1) Set the network variables.

2) Configure the decoder

3) Configure the base detection engine

4) Configure dynamic loaded libraries

5) Configure preprocessors

6) Configure output plugins

7) Customize your rule set

8) Customize preprocessor and decoder rule set

9) Customize shared object rule set

Conf example lines

```
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################
# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET
# List of ports you run web servers on
portvar HTTP_PORTS [36,80,81,82,83,84,85,86,87,88,89,90,311,383]
# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80
# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:
```

Conf example lines

```
##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################
# Stop generic decode events:
config disable_decode_alerts
# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts
# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts
# Stop Alerts on T/TCP alerts
config disable_tcpopt_ttcp_alerts
```

Conf example lines

```
##################################################

# Step #3: Configure the base detection engine.  For more information, see
README.decode

##################################################

# Configure PCRE match limitations

config pcre_match_limit: 3500

config pcre_match_limit_recursion: 1500

# Configure the detection engine  See the Snort Manual, Configuring Snort - Includes
- Config

config detection: search-method ac-split search-optimize max-pattern-len 20

# Configure the event queue.  For more information, see README.event_queue

config event_queue: max_queue 8 log 5 order_events content_length
```

Conf example lines

##

Step #4: Configure dynamic loaded libraries.

For more information, see Snort Manual, Configuring Snort - Dynamic Modules

##

path to dynamic preprocessor libraries

dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/

path to base preprocessor engine

dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

path to dynamic rules libraries

dynamicdetection directory /usr/local/lib/snort_dynamicrules

Conf example lines

```
##################################################

# Step #5: Configure preprocessors

# For more information, see the Snort Manual, Configuring Snort - Preprocessors

##################################################

# GTP Control Channle Preprocessor. For more information, see README.GTP

# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see README.normalize

# Target-based IP defragmentation.  For more inforation, see README.frag3

preprocessor frag3_global: max_frags 65536

preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10
min_fragment_length 100 timeout 180
```

Writing Snort rules

SNORT – MODES – PLUGINS - FLOW

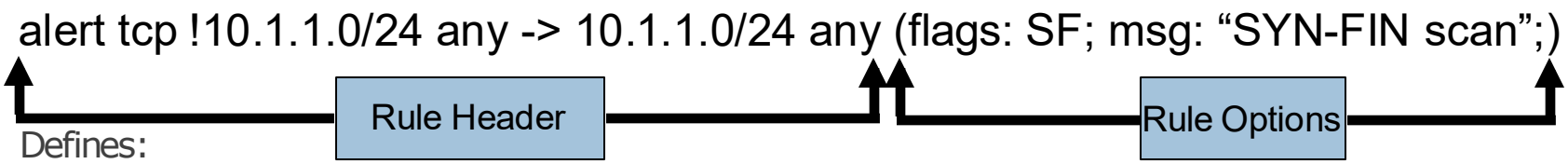
Snort Rules

- Rules define what Snort should watch for
  - Define 'who' and 'what' constitutes an attack or interesting activity
  - Simple syntax, easy to read and create
  - Inspect packet headers, payload or both
- Multi-packet attacks are usually detected by preprocessors
- https://snort.org/webcast_series

Rule characteristics

- Each rule has two parts
  - Rule header
  - Rule options
- Specific syntax for each part
- Rule header is required, rule options are not
- Rules may be on multiple lines if the "\" continuation character is used

Rule Header



- Defines:
  - Action (e.g. alert, pass, log, activate)
  - Protocol (e.g. tcp, ip, icmp, udp)
  - Source-destination IPs (CIDR notation)
    - any
    - IP(s) (e.g. 192.168.1.10/32, 192.168.1.0/24)
    - List (e.g. ![192.168.1.0/24,10.1.1.0/24])
    - Address exclusion (e.g. ![192.168.2.0/24])
  - Source-destination ports
    - any
    - Port number (e.g. 23, 80)
    - Port range (e.g. 1024:2048, 1024:, :1024)
    - Port exclusion (e.g. !53)
  - Direction of traffic (e.g. ->, <-, <>)

Snort manual, section 3.2, page 184

Action: The most common actions are listed below.

- **alert**: Generate an alert and log the packet.
- **log**: Log the packet.
- **drop**: Block and log the packet.
- **reject**: Block the packet, log it and terminate the packet session.

Protocol : Snort2 supports only four protocols filters in the rules (IP, TCP, UDP and ICMP). However, we can detect the application (like ftp, or ssh) flows using port numbers (21, 22, 23, etc)and options by investigating TCP traffic. Snort3 supports a wider range of protocols

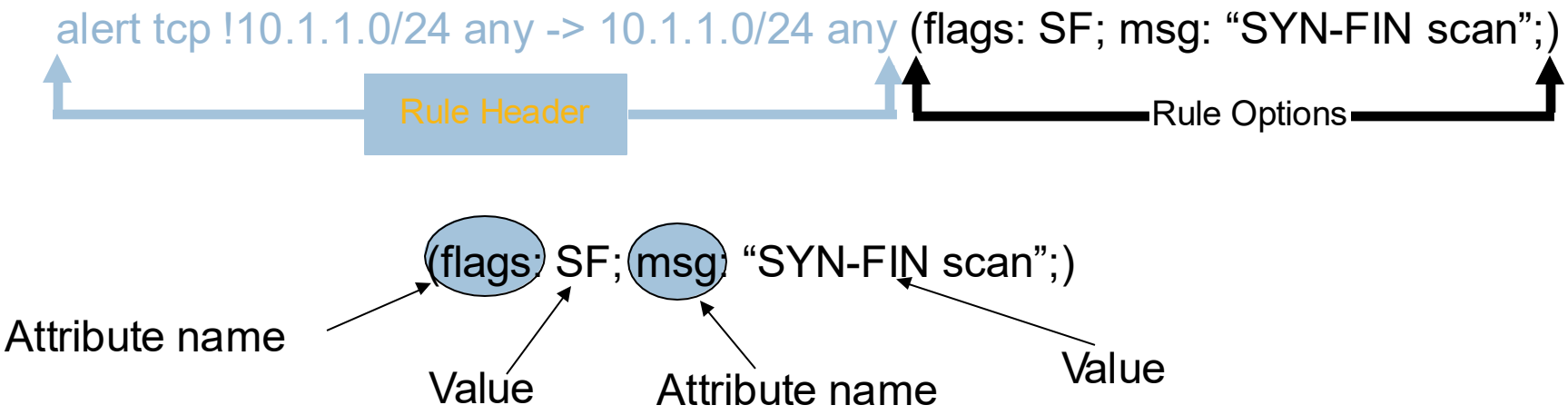
Exercise

Menti

Write rule header to generate an alert when detecting something over:

a. UDP connections
b. Icmp connections
c. Connections to databases, assuming the port of database server is 3306

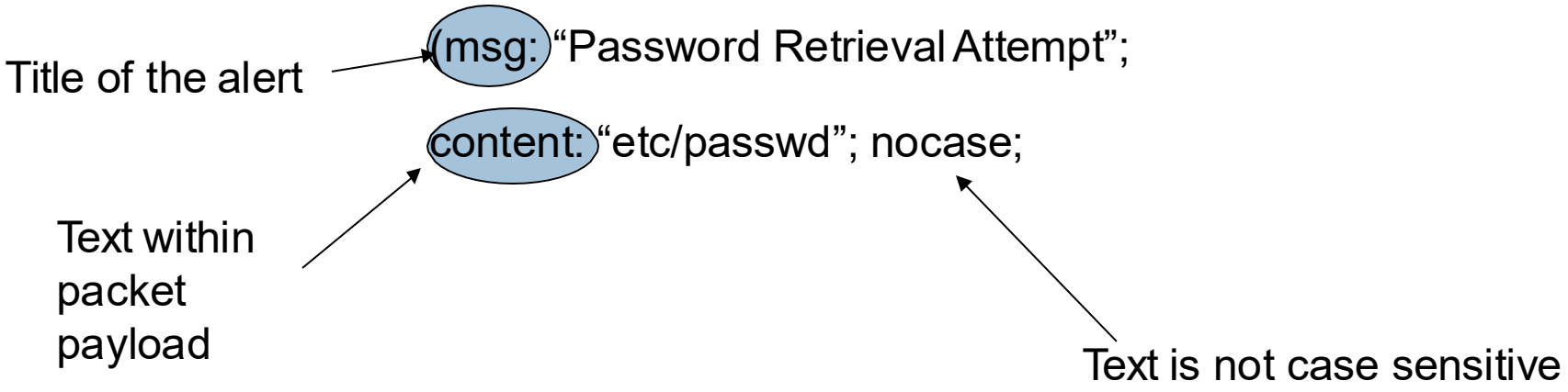
Rule Options



- Further define traffic/rule characteristics
  - can relate to header as well as payload
  - made up from individual packet attributes
- For more information on snort rules
  - https://snort.org/webcast_series

Rule Options

alert tcp !10.1.1.0/24 any -> 10.1.1.0/24 any (msg: “Password Retrieval Attempt”; content: “etc/passwd”; nocase;)



Rule options

- Event information
  - msg, sid, rev, reference, classification, priority
  - sid: 1-1,000,000 for snort.org, 2,000,000-3,000,000 for emerging threats, 3,000,000 - 0xFFFFFFFF for locally defined rules
- IP header inspection: Sameip, ip_proto, id, tos, ttl, fragbits, fragoffset
- TCP header inspection : flags, seq, ack, window, stateless, flow
- ICMP inspection: Itype, icode, icmp_id, icmp_seq
- Content inspection (Match data in the payload of a packet): content
- Active response: Used to trigger active responses when a rule matches
- Output modifiers: Can provide additional output for forensic analysis

Msg : Basic prompt and quick identifier of the rule. Once the rule is triggered, the message filed will appear in the console or log. Usually, the message part is a one-liner that summarises the event.

Reference : Each rule can have additional information or reference to explain the purpose of the rule or threat pattern. That could be a Common Vulnerabilities and Exposures (CVE) id or external information. Having references for the rules will always help analysts during the alert and incident investigation.

Rev : Refers to the revision number for the rule as rules can be modified and updated for performance and efficiency issues. It only indicates the number of times a rule has been modified. Rules have no history backup, so analyst should keep the history themselves. Each rule should have a unique rev number. *alert icmp any any <> any any (msg: "ICMP Packet Found"; sid: 100001; reference:cve,CVE-XXXX; rev:1;)*

ID — Filtering the IP id field.

Sameip — Filtering the source and destination IP addresses for duplication.

Flags — Filtering the TCP flags.

F — FIN ; S — SYN ; R — RST ; P — PSH; A — ACK; U — URG

Dsize — Filtering the packet payload size.

dsize:min<>max;

dsize:>100

dsize:<100

- **uricontent:** searches the NORMALIZED request URI field

  For example, the URI:

/scripts/..%c0%af../winnt/system32/cmd.exe?/c+ver

  will get normalized into: /winnt/system32/cmd.exe?/c+ver

- **flow**: check session properties of a given packet such as

  The direction of the packet, specifically whether it's from a client to a server or from a server to a client

  Whether the packet is part of an established TCP connection or not

  Whether the packet is a reassembled packet or not

  Whether the packet is a rebuilt frag packet or not

Argument	Description
`to_client`	Match on server responses
`to_server`	Match on client requests
`from_client`	Match on client requests
`from_server`	Match on server responses
`established`	Match only on established TCP connections
`not_established`	Match only when no TCP connection is established
`stateless`	Match regardless of stream state
`no_stream`	Match only on non-reassembled packets
`only_stream`	Match only on reassembled packets
`no_frag`	Match only on fragmented packets
`only_frag`	Match only on de-fragmented packets

Example:

flow:to_client,established;

Content

Content — Payload data. It matches specific payload data by ASCII, HEX or both. This rule option is case sensitive!

ASCII mode — alert tcp any any <> any 80 (msg: “GET Request Found”; content:”GET”; sid: 100001; rev:1;)

HEX mode — alert tcp any any <> any 80 (msg: “GET Request Found”; content:”|47 45 54|”; sid: 100001; rev:1;)

Content matching

- Multiple content modifier keywords. Specified after a particular content rule
  - **Offset**: Specifies byte offset from the *beginning of the payload*. Improves performance.
  - **Depth**: Specifies the amount of data from the initial byte offset to be searched in the packet. Improves performance.
  - **Nocase**: Disabling case sensitivity. Used for enhancing the content searches.
  - **Rawbytes** (in Snort2) or **raw_data** (in Snort3): inspect the raw, undecoded packet payload data, , ignoring any decoding that was done by preprocessors
  - **Distance:** Distance from the *end of last match* at which to start content check
  - **Within**: The amount of data from *the end of the last match* that the next match must occur within

Offset, Depth

alert tcp any any -> $HOME_NET 21 \
(content: "anonymous"; offset: 5; msg: "anonymous FTP login";)

The pattern matcher will start examining on the **byte offset** 5 (e.g. 6th byte) **of the payload** for the string "anonymous". If it matches, it will create an alert with title "anonymous FTP login"

alert tcp any any -> $HOME_NET 21 \
(content: "anonymous"; depth: 14; msg: "anonymous FTP login";)

The pattern matcher will search **the first** 14 byte of the payload for the string "anonymous". If it matches, it will create an alert with title "anonymous FTP login"

Exercise

Menti

Write snort rule options: move 2 bytes into the payload and look for “HTTP” within the next 8 bytes

Hint: offset starts counting from 0

Alert tcp any any -> any 80 (content:”HTTP”; offset: 2; depth: 8; msg:”found HTTP packet”;)

Distance, Within

```
alert tcp any any -> $HOME_NET 21 \
  (content: "USER"; content: "anonymous";\
   distance:1; msg: "Got anonymous FTP login";)
```

The pattern matcher will search for the string "USER" in packet payload. Then, it will start searching for the string "anonymous", at 1 byte **distance from the end of last content match**.

```
alert tcp any any -> $HOME_NET 110 \
  (content: "USER"; nocase; content: !"|0A|"; within: 256; \
   msg: "POP Username too long, possible overflow";)
```

The pattern matcher will search for the string "USER". Then if it doesn't find the hex code 0x0A **within the following** 256 bytes, it will create an alert with title "POP Username too long, possible overflow"

Exercise

Menti

Write snort rule options: Find “HTTP”. Then move 4 bytes from the end of “HTTP” and find “GET” within the next 5 bytes.

Hint: offset starts counting from 0

Rawbytes, Uricontent

```
alert tcp any any -> any 21 \
    (msg:"Telnet NOP"; content:"|FF F1|"; rawbytes;)
```

Preprocessor often normalise traffic. The pattern matcher will search for hexadecimal content “FF F1” in the raw packet payload (un-normalised traffic). Rawbytes is useful for forensics, and finding shellcode.

```
alert tcp any any -> $HTTP_SERVERS 80 \
   (flow:to_server,established; uricontent:”/chatreader.php”; nocase;\
   msg: “WEB-PHP myPHPNuke chatreader.php access”;)
```

Uricontent works with the http decode preprocessor. The pattern matcher will search for the string “/chatreader.php” in the normalised web traffic.

Configuration and Deployment

SCENARIOS - PERFORMANCE

Snort Performance

- Snort can be configured in very high performance run-time mode
  - Important in high-speed networks
- Snort can also be "detuned" to be extremely slow
  - Provides more information about events
- Why is performance critical?
  - Bad performance = false negatives

Snort: Running Slow

- Run in verbose/data dump mode
- Log to ASCII files
- Load up more than a few output plug-ins
- Misconfigure preprocessors
- Run too many rules
- Slow hardware
- Excessive session capture

Snort: Running fast

- Binary mode logging
- Reduced rule set
- Conservative settings for preprocessors
- Limited number of output plugins
- No screen printing, no ASCII logging
- Example configuration
  - log_tcpdump
  - alert_fast or alert_syslog
  - Stream preprocessor set to minimum port set and memory footprint
  - Portscan preprocessor set for short detection periods

Thank You

NEXT LECTURE WILL BE ON NETWORK TRAFFIC