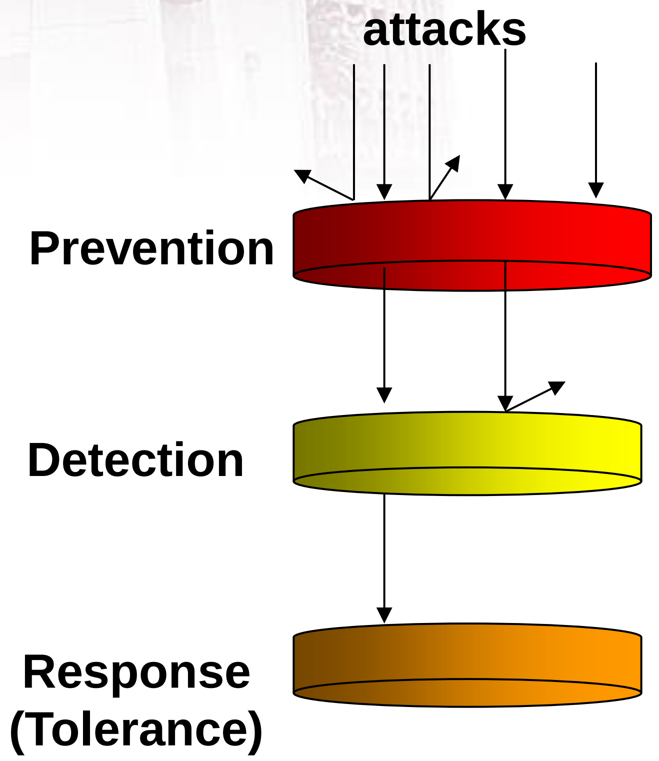
Network Intrusion Detection System
A case study through snort

Chun-Jen (James) Chung

Arizona State University

Security mechanisms

- In general, three types
  - Prevention
    - Example: access control (e.g., firewall)
  - Detection
    - Example: Auditing and intrusion detection (e.g., IDS, forensics)
  - Tolerance
    - Example: intrusion tolerance (e.g., ITS)



IDS and Snort

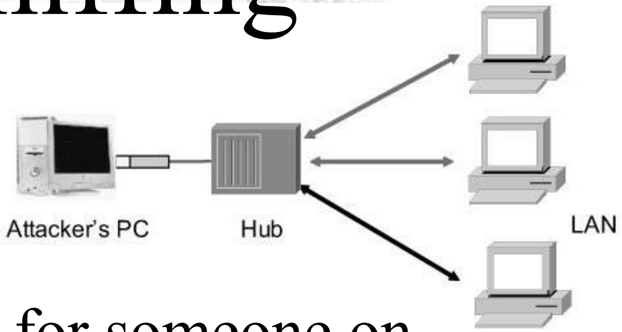
- Intrusion Detection System (IDS)
  - – IDS is software, hardware or combination of both used to *detect intruder activity*.
- Snort is an open source IDS
  - It is a multi-mode packet analysis tool
  - Sniffer (Passive and Active Sniffer)
    - » Port mirror sniffer, GW sniffer
  - Packet Logger
  - Data Analysis tool
  - Network Intrusion Detection System

Sniffing and Sniffer

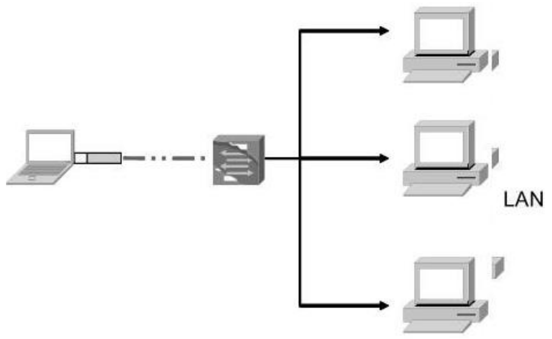
Sniffing

- Sniffing is a electronic form of eavesdropping on the communications that computers transmit across networks.
- ***Sniffers*** (a powerful piece of software) place the hosting system's network card into ***promiscuous*** mode
  - to receive all the data it can see, not just packets addressed to it.
- Sniffer peels away the layers of encapsulation and decoded the relevant information in the packet.
- What protocols are vulnerable to sniffing?

Active vs. Passive Sniffing



- Passive sniffing is performed on a hub
  - All traffic is sent to all ports.
  - Attacker (*without injects packets*) just wait for someone on the same **collision domain** to start sending or receiving data.
- Active sniffing is performed on a switched network
  - Relies on *injecting packets (probes*) into the network that causes traffic.
  - It is required to bypass the segmentation that switches provided.



Collision vs. Broadcast Domains

- ***Collision domain***
  - A logical area of the network in which one or more data packets can collide with each other.
  - found in the hub or other shared medium networks, e. g. wireless network such as Wi-Fi.
  - Modern wired networks use a switch to eliminate collisions.
- ***Broadcast Domain***
  - a logical division of a network, in which all nodes can reach each other by broadcast at the data link layer. A broadcast domain can be within the same LAN segment or it can be bridged to other LAN segments.

Protecting Against Sniffing

- To protect wired/wireless users from sniffing is to utilize encrypted sessions wherever possible:
  - SSL for e-mail connection
  - SSH instead of Telnet
  - SCP instead of FTP
- To protect a network from being discovered with sniffing tools
  - turn off any network identification broadcasts
  - if possible, close down the network to any unauthorized users.

Category of IDS

Category of IDS (by functionality)

- Network Intrusion Detection System (NIDS)
  - Listens & analyses traffic in a network
  - Capture data package
  - Compare with database signatures (signature-based)
  - Operate in promiscuous mode
- Host-based Intrusion Detection System (HIDS)
  - Installed as an agent of a host
  - Listens & analyses system logs

Type of IDS (by detection)

- Signature-based IDS
  - Captures and monitors packets in a network
  - Compares them with pre-configured and pre-determined attack patterns (signatures)
- Anomaly-based IDS
  - Determines what is normal network activities
    - Bandwidth, protocols, ports and devices
  - Raises alerts when anomalous traffic is detected
    - e.g. suddenly increase in TCP connection from 20 to 2000

Snort Basic

Snort Features

- *Compact*
  - The source is 5MBs version 2.9.6.0
- *Portable*
  - Linux, Windows, MacOS X, Solaris, BSD, etc.
- *Fast*
  - High probability of detection for a given attack on Gigabit networks
- *Configurable*
  - Easy rules language, many reporting/logging options
- *Used for Free*
  - GPL/Open Source Software (An alternate: OpenVAS)

Required Software (Windows)

- *WinPcap*
  - WinPcap is a packet capturing library in Windows.
- *Barnyard*
  - Barnyard is an output system for Snort.
  - Snort creates a special binary output format called **unified**.
  - Barnyard reads this file, and then resends the data to a database backend.

Required Software (Linux)

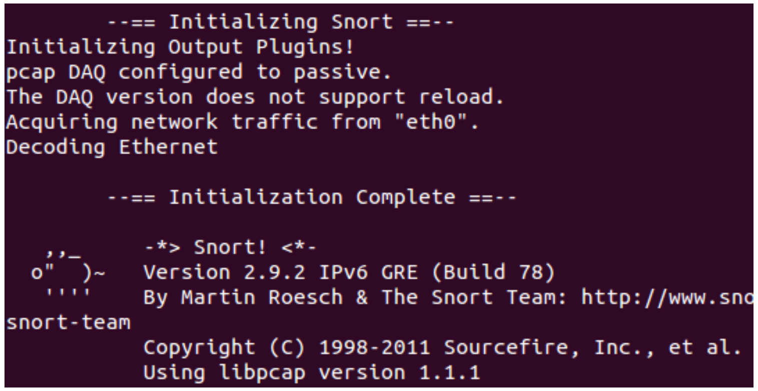
- *Libpcap*
  - pcap (packet capture) consists of an application programming interface (API) for capturing network traffic.
- *PCRE*
  - Perl Compatible Regular Expressions (PCRE) is a regular expression C library inspired by Perl's external interface.
  - The PCRE library is incorporated into a number of prominent open-source programs such as the Apache HTTP Server, the PHP and R scripting languages, and Snort.

Required Software (Linux)

- *Libdnet*
  - Libdnet is a generic networking API that provides access to several protocols.
- DAQ
  - DAQ is the Data-Acquisition API that replaces direct calls into packet capture libraries like PCAP,PFPACKET, NFQ, IPFW with an abstraction layer that make it easy to add additional software or hardware packet capture implementations.
- *Barnyard2*
  - The output process plug-ins.

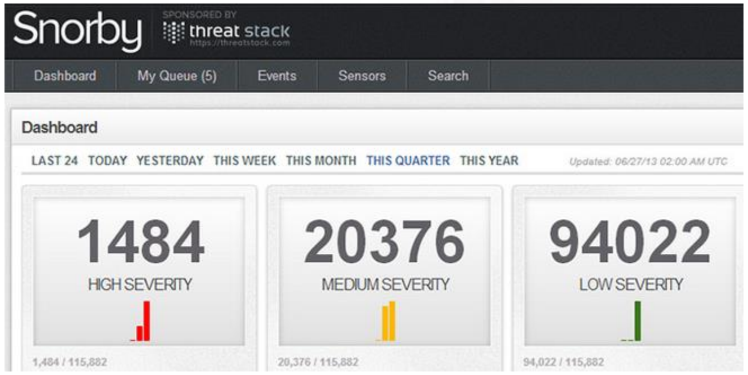
Snort Related Software Packages

Snort (intrusion detection system)

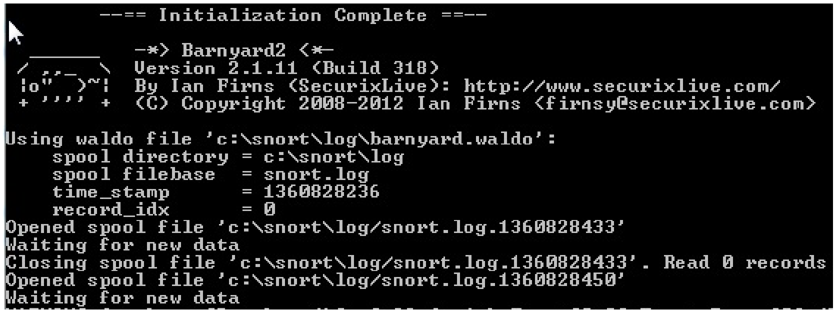


```
        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.2 IPv6 GRE (Build 78)
   ''''    By Martin Roesch & The Snort Team: http://www.sno
snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
```

Snorby (reporting system)

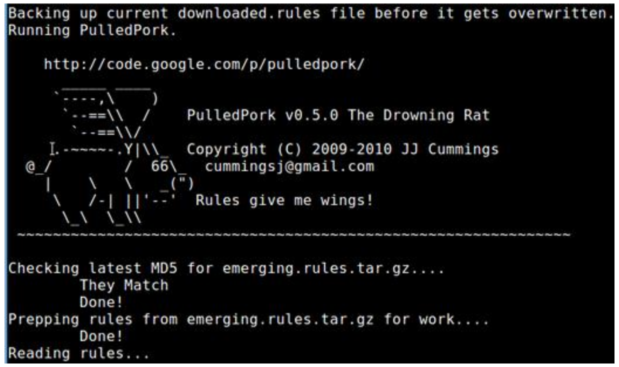


Banyard2 (output plug-ins)



```
        --== Initialization Complete ==--

   ______       -*> Barnyard2 <*-
  / ,,_  \      Version 2.1.11 (Build 318)
  |o"  )~|      By Ian Firns (SecurixLive): http://www.securixlive.com/
  + '''' +      (C) Copyright 2008-2012 Ian Firns <firnsy@securixlive.com>

Using waldo file 'c:\snort\log\barnyard.waldo':
    spool directory = c:\snort\log
    spool filebase  = snort.log
    time_stamp      = 1360828236
    record_idx      = 0
Opened spool file 'c:\snort\log/snort.log.1360828433'
Waiting for new data
Closing spool file 'c:\snort\log/snort.log.1360828433'. Read 0 records
Opened spool file 'c:\snort\log/snort.log.1360828450'
Waiting for new data
```

PulledPork (rules management)



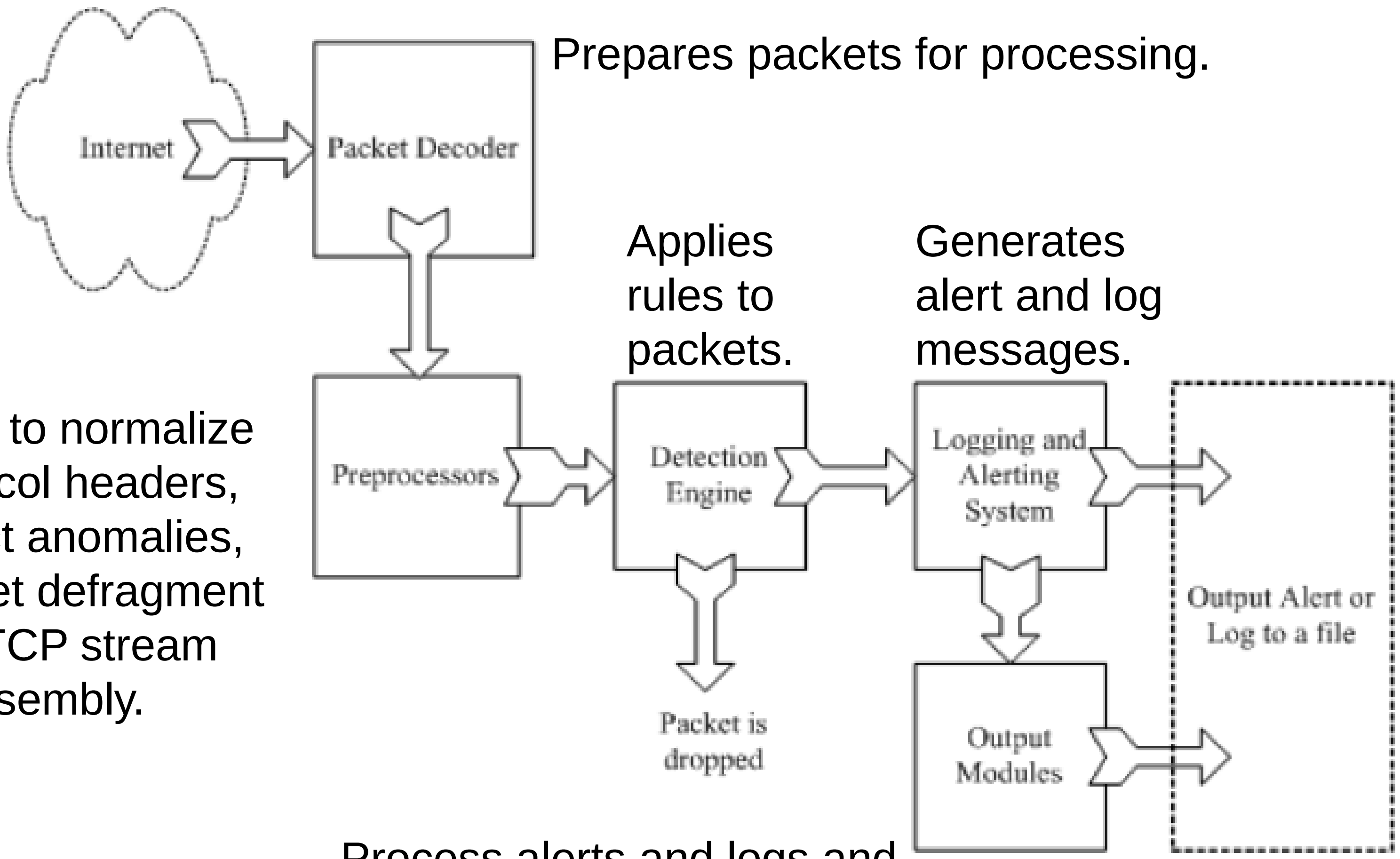
```
Backing up current downloaded.rules file before it gets overwritten.
Running PulledPork.

    http://code.google.com/p/pulledpork/

     `----,\    )
      `--==\\  /     PulledPork v0.5.0 The Drowning Rat
       `--==\\/
    I.~~~~~.Y|\\_    Copyright (C) 2009-2010 JJ Cummings
  @_/        /  66\_   cummingsj@gmail.com
    |    \   \   _(")
     \   /-| ||'--'   Rules give me wings!
      \_\  \_\\
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Checking latest MD5 for emerging.rules.tar.gz....
        They Match
        Done!
Prepping rules from emerging.rules.tar.gz for work....
        Done!
Reading rules...
```

Snort Architecture

Snort Design

- Packet sniffing “lightweight” network intrusion detection system
- Libpcap-based sniffing interface
- Rules-based detection engine
- Plug-ins system allows endless flexibility

Snort Components



Prepares packets for processing.

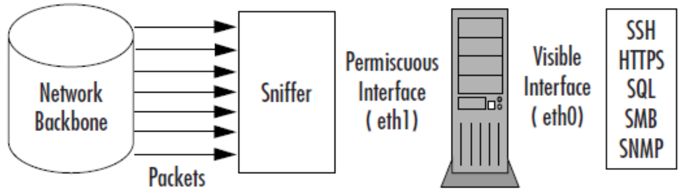
Applies rules to packets.

Generates alert and log messages.

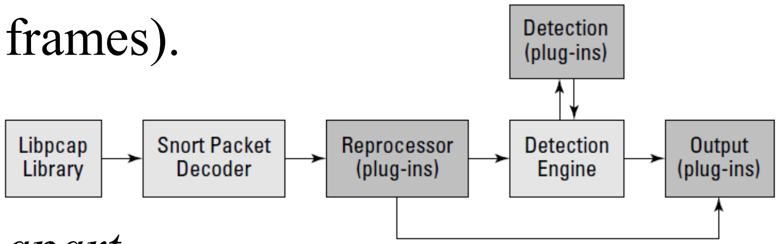
Used to normalize protocol headers, detect anomalies, packet defragment and TCP stream re-assembly.

Process alerts and logs and generate final output.

Packet Sniffer, Capture and Decoder

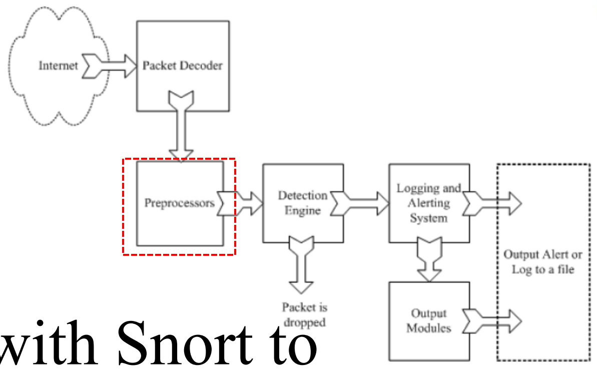


- **Packet Sniffer.** Snort enables the promiscuous mode in NICs and makes them as packet sniffers used to tap into networks.
- **Packet capture library** is a separate piece of software that tosses Snort network packets from the network card. There are unprocessed layer 2 packets (Ethernet frames).
  – *Libpcap* in linux, *WinPcap* in Windows systems.

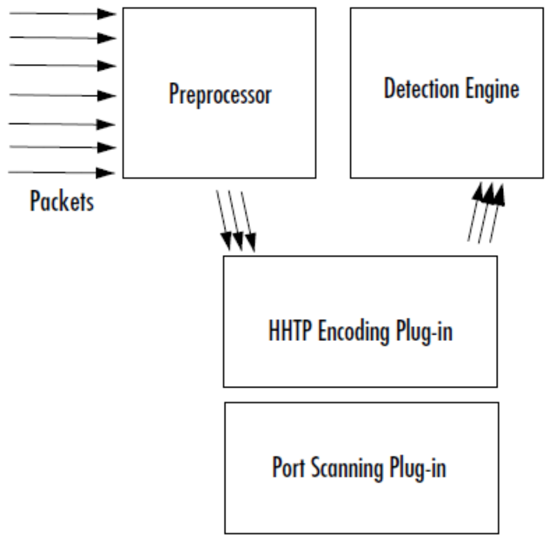


- **Packet decoder** takes the Layer 2 data sent over from the packet capture library and *takes it apart*.
  – First it decodes the Data Link frame, then the IP protocol, then the TCP or UDP packet.
  – When finished decoding, Snort has all the protocols information in all the right places for further processing.

Preprocessors



- Preprocessors are
  - components or plug-ins that can be used with Snort to *arrange or modify data packets* before the detection engine does some operation.
  - Plug-ins can be enabled and disabled.
- The preprocessor also performs
  - Finding anomalies in packet headers
  - Packet defragmentation
  - Decode HTTP URI
  - Re-assemble TCP streams

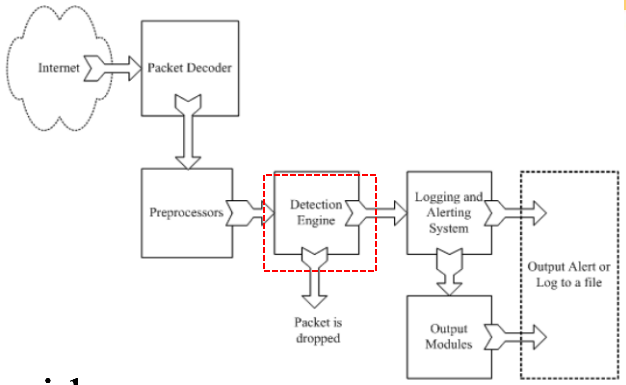


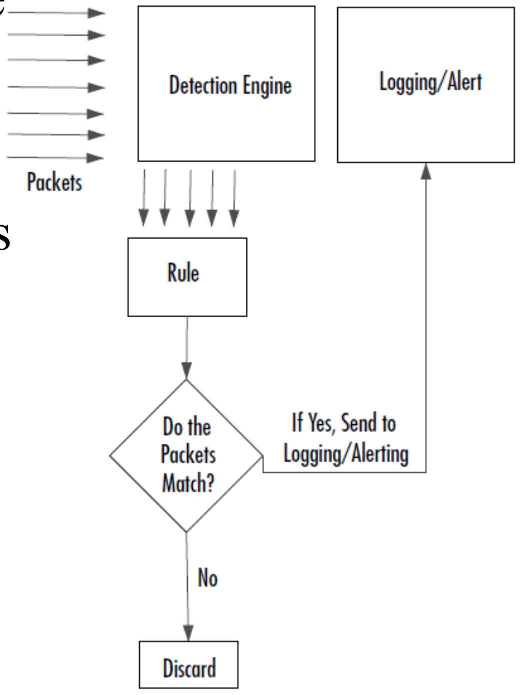
Available Preprocessors in Snort

Preprocessor name	Function
Flow	This preprocessor helps keep a state flow log of packets passing through the Snort engine.
Frag2	This preprocessor detects and reassembles fragmented packets attempting to bypass detection.
stream4	This preprocessor reassembles TCP packets and inspects them to detect attempted IDS evasion attacks from tools such as snot or stick using stateless attacks.
stream4_reassemble	This is the second part of the stream4 engine. It reassembles packets into meaningful sessions for the Snort rules engine and for the preprocessors loaded after Snort.
Http_inspect	This is a new preprocessor that handles all HTTP traffic to help speed it through to the rules engine.
rpc_decode	This is actually only an application decoder. It listens for RPC protocol packets on certain ports, and then decodes the traffic on those ports to ASCII to be passed back to the Snort rules engine for comparison.
telnet_decode	This is also an application decoder. It decodes all traffic on several ports, including 23/tcp, and then passes it back to the Snort engine.
bo_decode	This preprocessor detects when the popular Trojan horse program Back Orifice is in use on your network. This highly popular Trojan has its own protocol that Snort is able to quickly detect and pass on to the rules engine for detailed inspection to determine the commands in use. Subseven and several other Trojan tools have surpassed this Trojan.
Flow-portscan	This is the only preprocessor that has to have the flow preprocessor enabled to work. It takes flow data and finds the port scans in that data.
Arpspoof	This preprocessor is fed a list of IP:MAC addresses. When it detects a layer-2 attack, it triggers an alarm for a layer-2 event, such as multiple MAC addresses from a single IP.
Perfmonitor	This is a new preprocessor that generates statistical information on the load Snort is under, sensor load, and several network performance measurements.

Detection Engine

- The kernel part of Snort to perform:
  - *Rule Parsing*
    - Rules are loaded into internal data structures, and guide packet inspection
  - *Signature Detection*
    - Attack signatures are constructed by parsing Snort rules
    - Detect if any intrusion activity exists in a packet by checking the rule set.



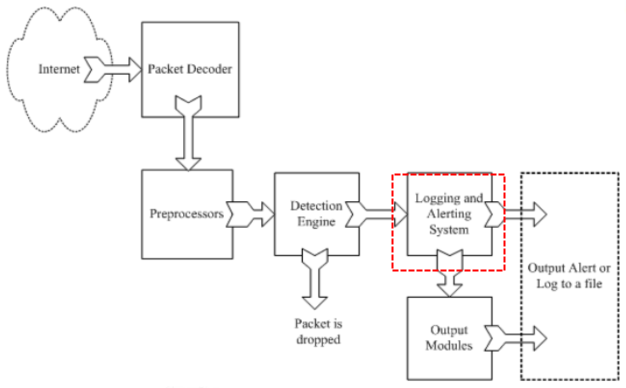


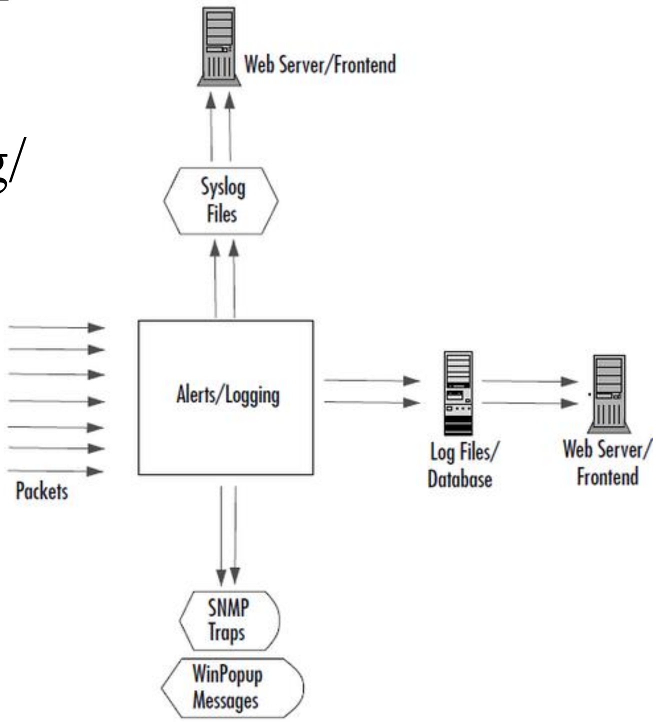
Detection Engine (cont.)

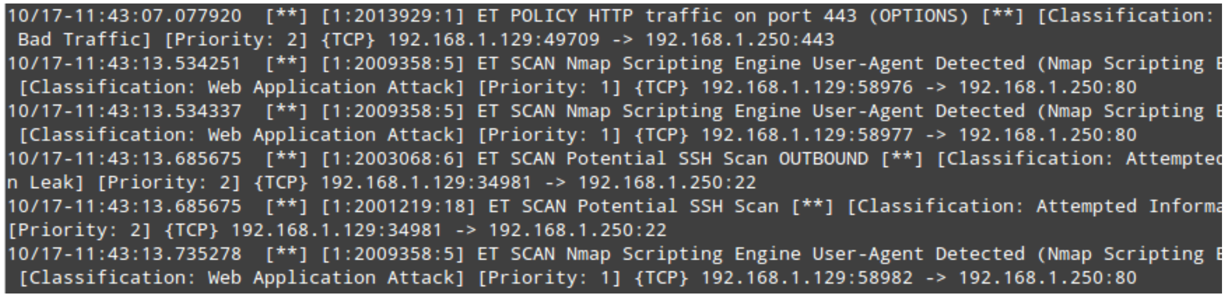
- The time-critical part of Snort.
  - The NIDS mode in Snort will drop packets if there are too many rules or traffic to be checked.
- The load on the detection engine depends on:
  - Number of rules, Power of the machine, Speed of the internal bus, Load on the network
- The detection system can dissect a packet and apply rules on different parts of the packet.
  - IP header
  - Transport layer header: TCP, UDP, ICMP header or other transport layer headers.
  - Application layer level header: DNS, FTP, SNMP, SMTP
  - Packet payload: Self-defined rule to find particular string in side the packet.

Logging and Alerting System

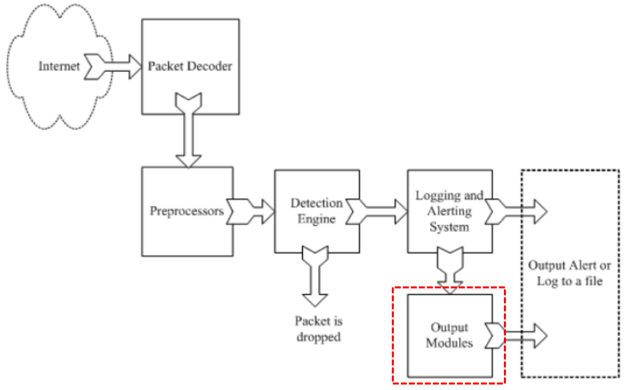
- Raise an alert or log the activity based on the decision from the detection engine.
- Logs are kept in simple text files, tcpdump-style files or some other form.
- All of the log files are stored under /var/log/snort folder by default.






```
10/17-11:43:07.077920  [**] [1:2013929:1] ET POLICY HTTP traffic on port 443 (OPTIONS) [**] [Classification:
 Bad Traffic] [Priority: 2] {TCP} 192.168.1.129:49709 -> 192.168.1.250:443
10/17-11:43:13.534251  [**] [1:2009358:5] ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting E
 [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.1.129:58976 -> 192.168.1.250:80
10/17-11:43:13.534337  [**] [1:2009358:5] ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting E
 [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.1.129:58977 -> 192.168.1.250:80
10/17-11:43:13.685675  [**] [1:2003068:6] ET SCAN Potential SSH Scan OUTBOUND [**] [Classification: Attempted
n Leak] [Priority: 2] {TCP} 192.168.1.129:34981 -> 192.168.1.250:22
10/17-11:43:13.685675  [**] [1:2001219:18] ET SCAN Potential SSH Scan [**] [Classification: Attempted Informa
[Priority: 2] {TCP} 192.168.1.129:34981 -> 192.168.1.250:22
10/17-11:43:13.735278  [**] [1:2009358:5] ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting E
 [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.1.129:58982 -> 192.168.1.250:80
```

Output Modules



- Control the type of output generated by the logging and alerting system.
- Depending on the configuration, output modules can do things like the following:
  - Simply logging to /var/log/snort/alerts file or some other file
  - Sending SNMP traps
  - Sending messages to syslog facility
  - Logging to a database like MySQL or other database
  - Generating eXtensible Markup Language (XML) output
  - Modifying configuration on routers and firewalls.

IDS vs. IPS

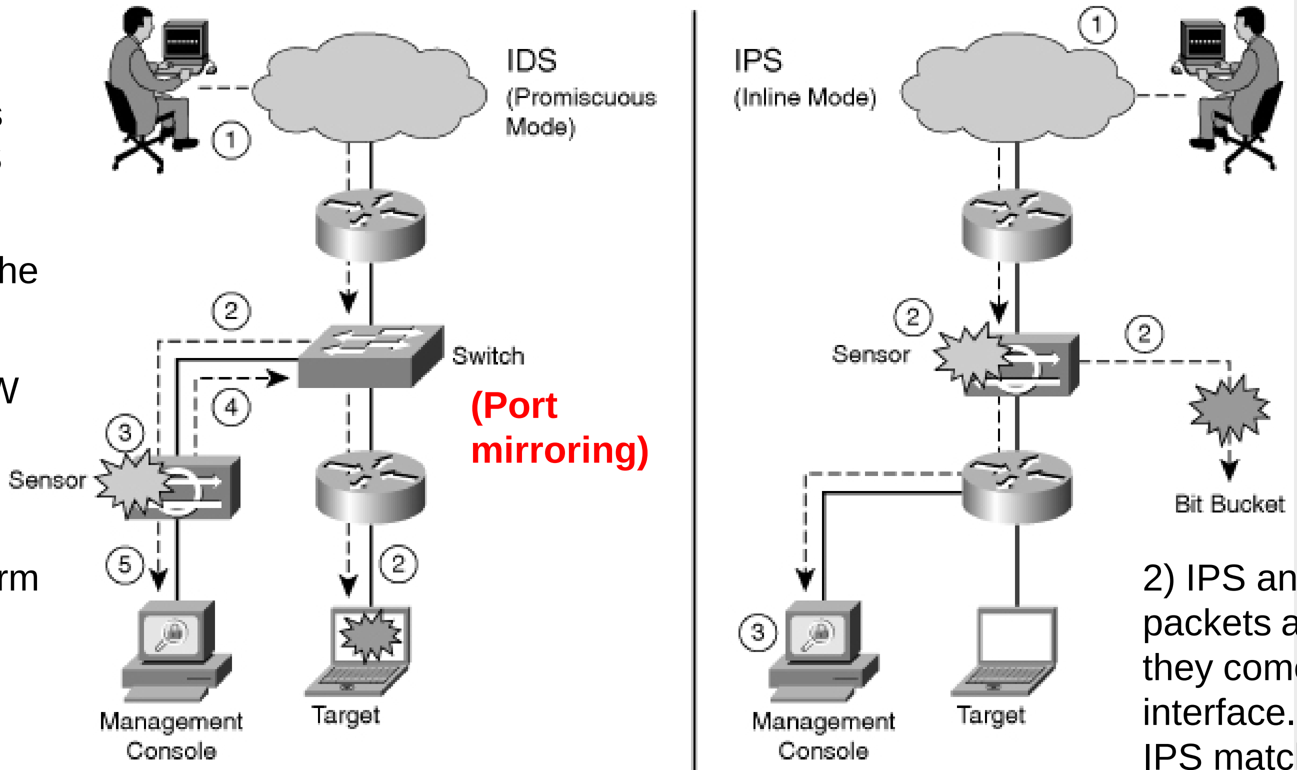
1) An attack is launched

2) SW sends copies of all packets to IDS

3) IDS matches the malicious traffic to the signature

4) IDS sends the SW a command to deny the traffic

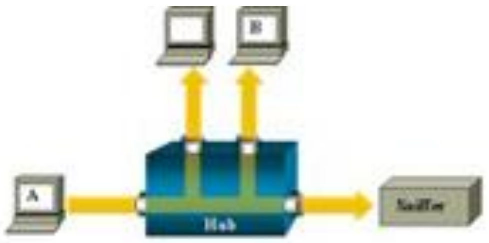
5) IDS sends an alarm to a management console



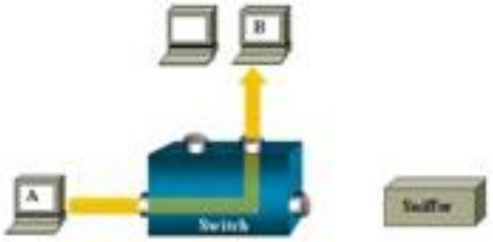
2) The target machine experiences the malicious attack

2) IPS analyzes the packets as soon as they come into the IPS interface.
IPS matches the malicious traffic to the signature and stops the attack immediately.

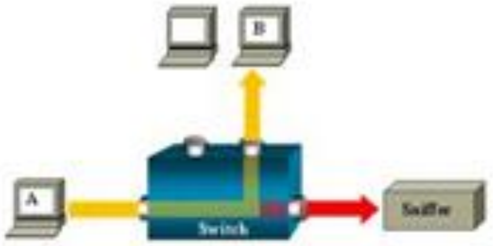
Port Mirroring (SPAN)



In the past, with a shared hub, packets are replicated on all ports; plug a "sniffer" into any data port will see all traffic!!



With a LAN switch, traffic travels point-to-point only; packets no longer replicated; network visibility is lost



SPAN (or mirroring) port was invented to replicate packets of a single port or a single VLAN for monitoring

Pros and Cons of IDS and IPS

- IDS

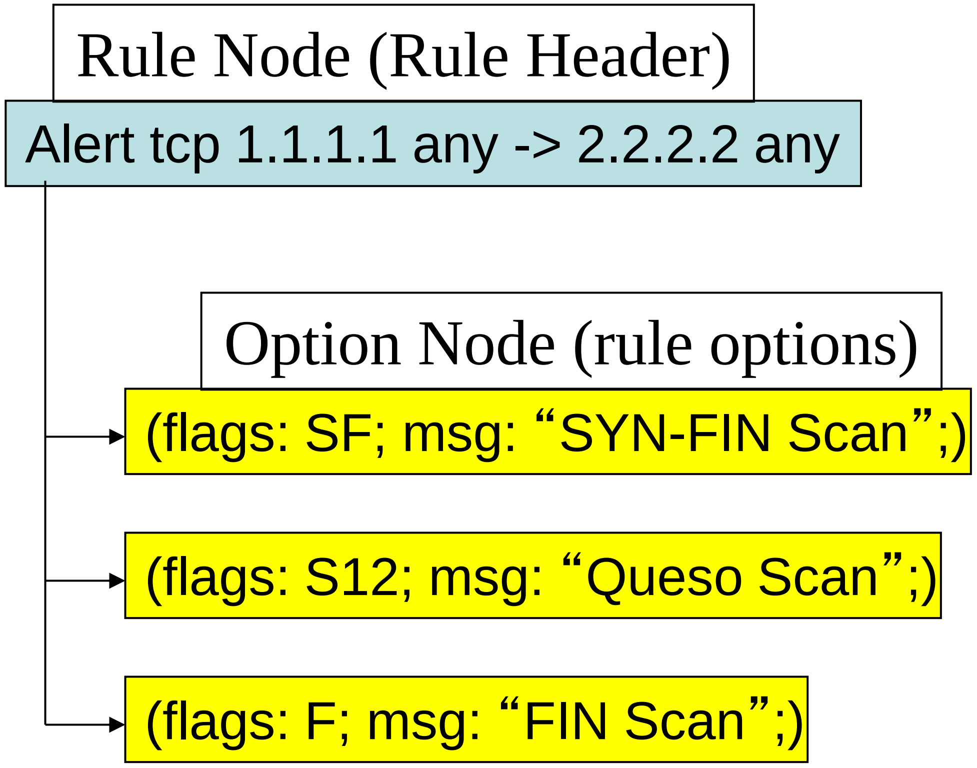
Advantages	**Limitations**
No impact on network (latency, jitter)	Response action cannot stop trigger packets
No impact if there is a sensor failure	Correct tuning required for response actions
No impact if there is a sensor overload	Must have a well-thought out security policy
	More vulnerable to network evasion techniques

- IPS

Advantages	**Limitations**
Stops trigger packets	Sensor issues might affect network traffic
Can use stream normalization techniques	Sensor overloading impacts the network
	Must have a well-thought out security policy
	Some impact on network (latency, jitter)

Internal Representation for the rule set

- There are five separate chains of rules.
  - **Activation:** Alert and then turn on another dynamic rule.
  - **Dynamic:** Log the traffic when called by the above activation rule.
  - **Alert:** Generate an alert and then log the packet.
  - **Pass:** Ignore this packet.
  - **Log:** Log the traffic (don't alert).

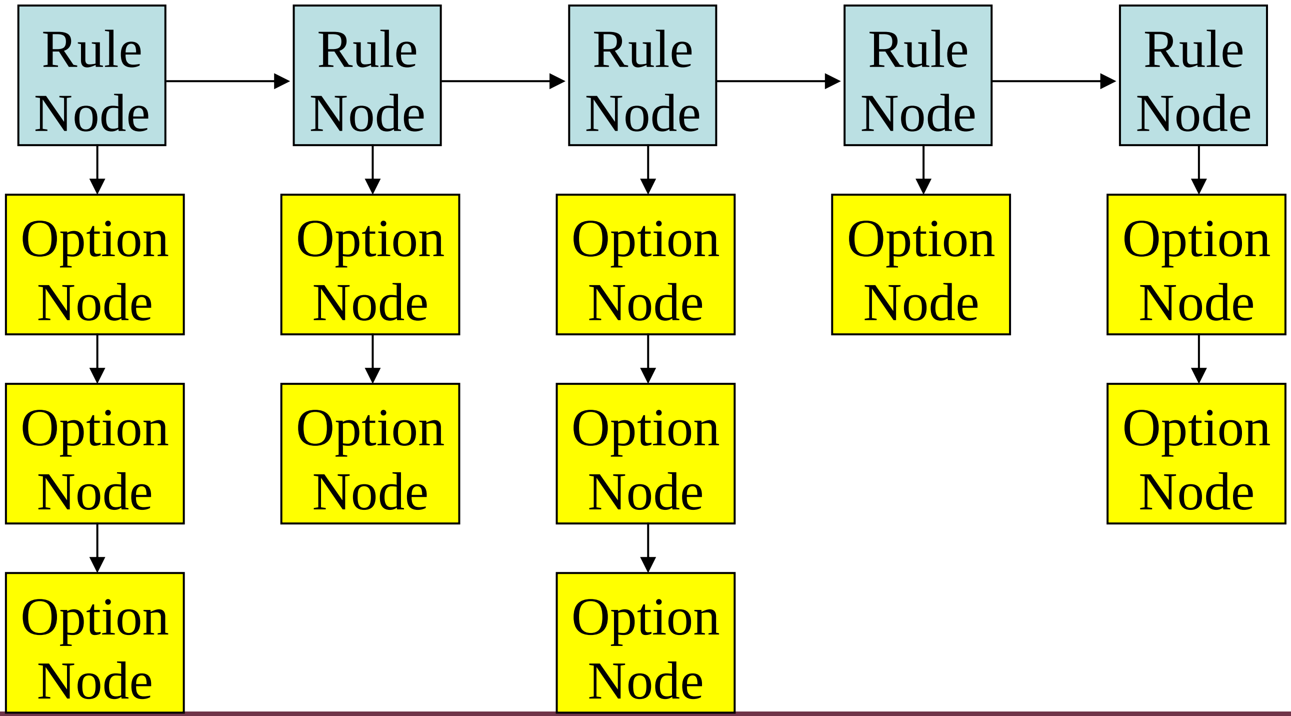

Rule Node (Rule Header)
Alert tcp 1.1.1.1 any -> 2.2.2.2 any
Option Node (rule options)
(flags: SF; msg: "SYN-FIN Scan";)
(flags: S12; msg: "Queso Scan";)
(flags: F; msg: "FIN Scan";)

Internal Representation for the rule set

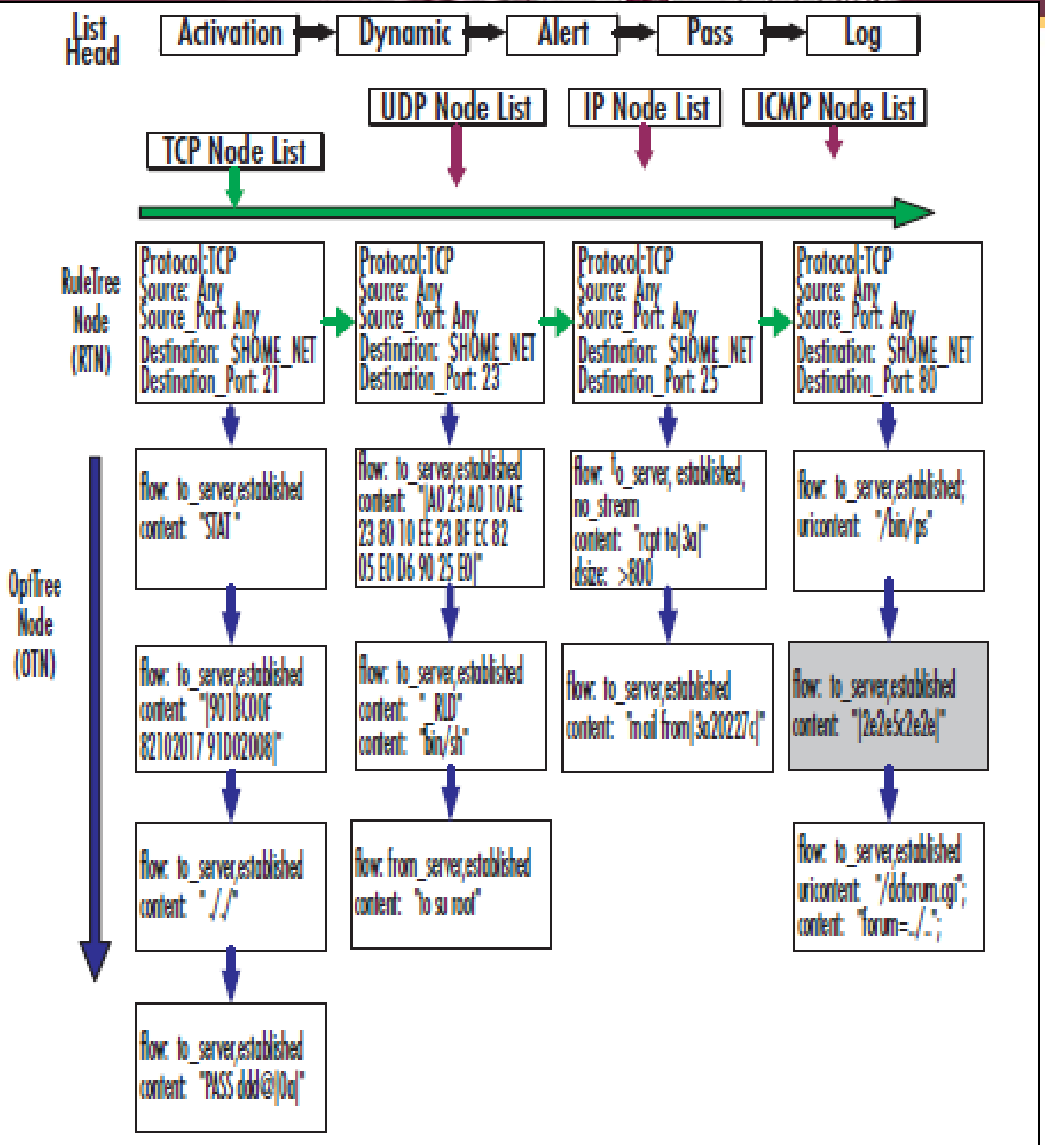
- For each of the five chains of rules, there are separate linked lists, broken down by protocol. This level of the tree is referred to as the ***Rule Tree Nodes (RTN)***:
  - **TCP:** TCP protocol; for example, *SMTP, HTTP, FTP*
  - **UDP:** UDP protocol; for example, *DNS lookups*
  - **ICMP:** ICMP protocol; for example, *ping, traceroute*
  - **IP:** IP protocol; for example, *IPSec, IGMP*
- Within each of the protocol linked lists are the Rule options, ***Option Tree Nodes (OTN)***. E.g.:
  - **Content:** Content checked by the Boyer-Moore pattern matching algorithm
  - **Flow:** Link to detection plug-in
- On initialization, Snort reads the rule files and populates the linked lists.

Internal data structure for the rule set

- Snort uses a 3D linked list to store rules and their options and then searches the list for a rule header match.
- Then, within the header match, it searches for a pattern match or a match using a detection plug-in.



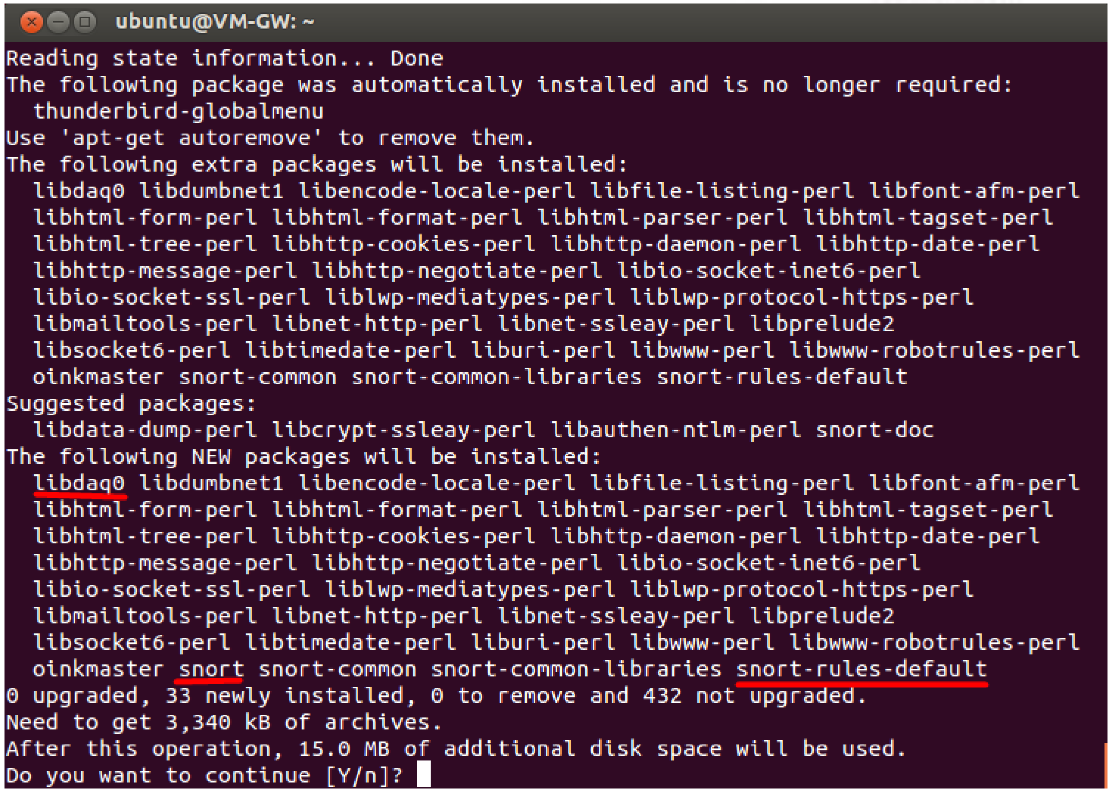
- The figure shows a populated *OptionTreeNode* for the *TCP RuleTreeNode* off the Alert chain.


List Head
Activation
Dynamic
Alert
Pass
Log
UDP Node List
IP Node List
ICMP Node List
TCP Node List
RuleTree Node (RTN)
Protocol: TCP Source: Any Source_Port: Any Destination: $HOME_NET Destination_Port: 21
Protocol: TCP Source: Any Source_Port: Any Destination: $HOME_NET Destination_Port: 23
Protocol: TCP Source: Any Source_Port: Any Destination: $HOME_NET Destination_Port: 25
Protocol: TCP Source: Any Source_Port: Any Destination: $HOME_NET Destination_Port: 80
OptTree Node (OTN)
flow: to_server,established content: "STAT"
flow: to_server,established content: "|A0 23 A0 10 AE 23 80 10 EE 23 BF EC 82 05 E0 D6 90 25 E0|"
flow: to_server, established, no_stream content: "rcpt to|3a|" dsize: >800
flow: to_server,established; uricontent: "/bin/ps"
flow: to_server,established content: "|90 1BC0 0F 82102017 91D02008|"
flow: to_server,established content: " RLD" content: "bin/sh"
flow: to_server,established content: "mail from|3a20227c|"
flow: to_server,established content: "|2e2e5c2e2e|"
flow: to_server,established content: "././"
flow: from_server,established content: "to su root"
flow: to_server,established uricontent: "/dcforum.cgi"; content: "forum=../..";
flow: to_server,established content: "PASS ddd@|0a|"

Snort
Installation and Configuration

Installation

- sudo apt-get install snort

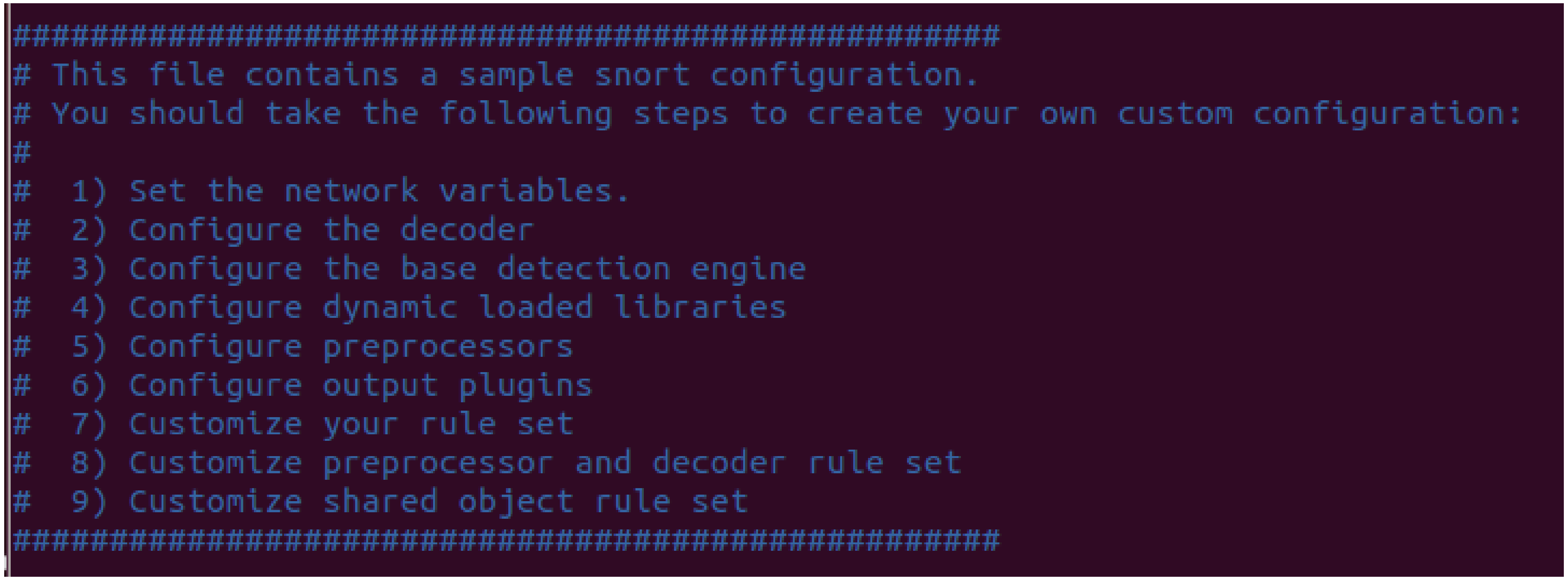


```
Reading state information... Done
The following package was automatically installed and is no longer required:
  thunderbird-globalmenu
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  libdaq0 libdumbnet1 libencode-locale-perl libfile-listing-perl libfont-afm-perl
  libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-tree-perl libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
  libhttp-message-perl libhttp-negotiate-perl libio-socket-inet6-perl
  libio-socket-ssl-perl liblwp-mediatypes-perl liblwp-protocol-https-perl
  libmailtools-perl libnet-http-perl libnet-ssleay-perl libprelude2
  libsocket6-perl libtimedate-perl liburi-perl libwww-perl libwww-robotrules-perl
  oinkmaster snort-common snort-common-libraries snort-rules-default
Suggested packages:
  libdata-dump-perl libcrypt-ssleay-perl libauthen-ntlm-perl snort-doc
The following NEW packages will be installed:
  libdaq0 libdumbnet1 libencode-locale-perl libfile-listing-perl libfont-afm-perl
  libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-tree-perl libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
  libhttp-message-perl libhttp-negotiate-perl libio-socket-inet6-perl
  libio-socket-ssl-perl liblwp-mediatypes-perl liblwp-protocol-https-perl
  libmailtools-perl libnet-http-perl libnet-ssleay-perl libprelude2
  libsocket6-perl libtimedate-perl liburi-perl libwww-perl libwww-robotrules-perl
  oinkmaster snort snort-common snort-common-libraries snort-rules-default
0 upgraded, 33 newly installed, 0 to remove and 432 not upgraded.
Need to get 3,340 kB of archives.
After this operation, 15.0 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

- sudo apt-get install snort-mysql

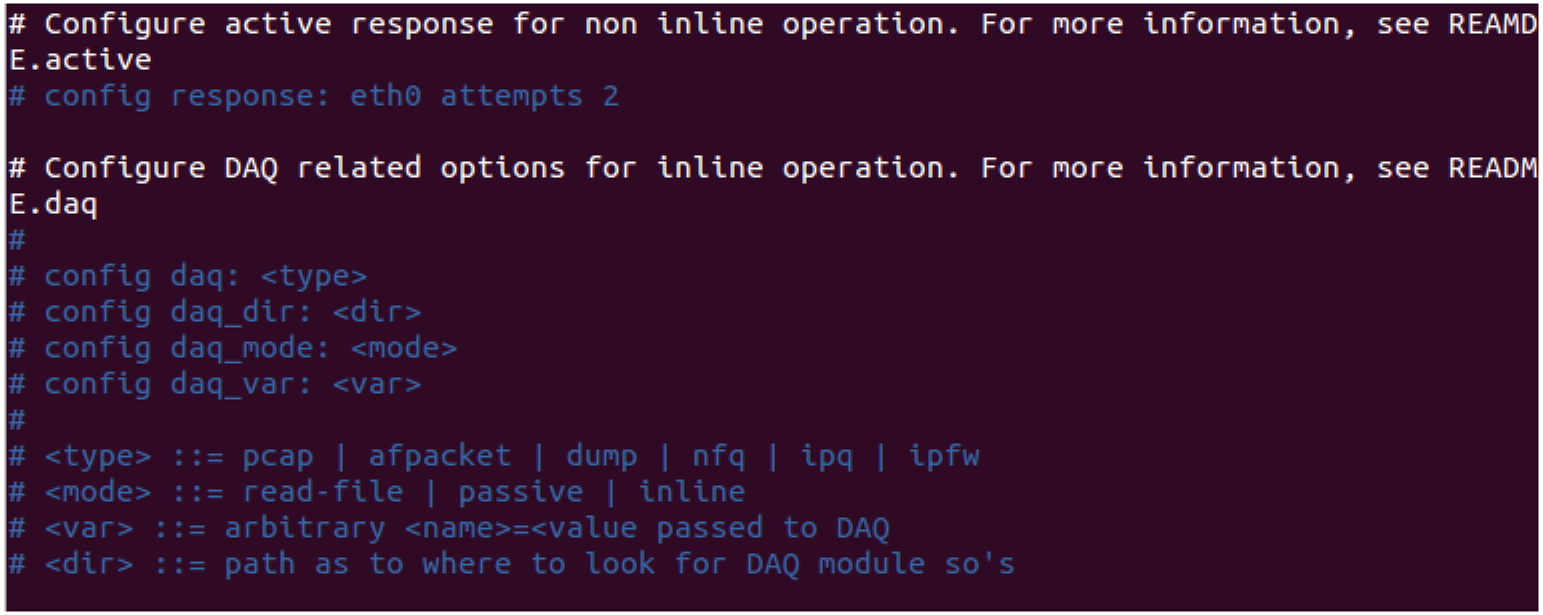
Configuration

- /etc/snort.conf



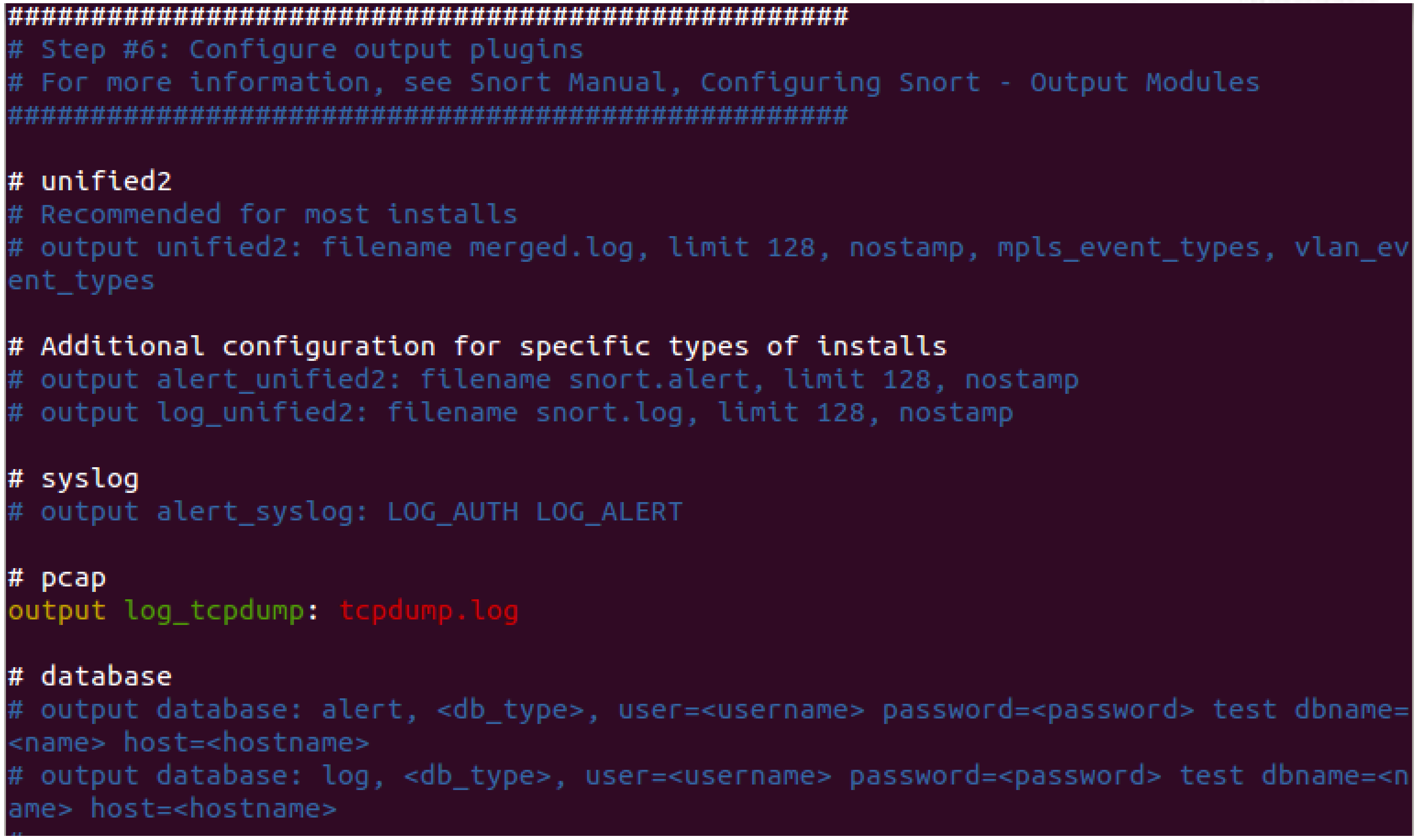
```
###################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure 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
###################################################
```

Configure DAQ



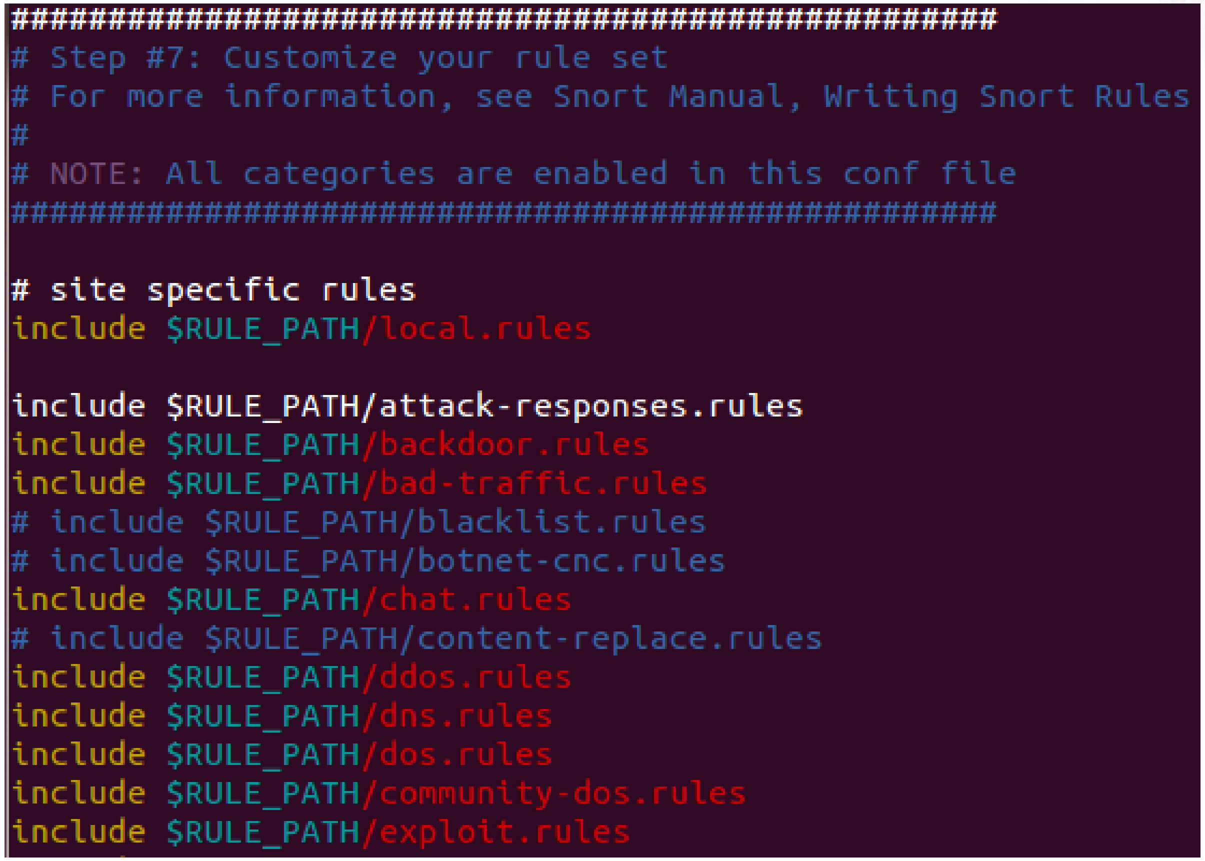
```
# Configure active response for non inline operation. For more information, see READM
E.active
# config response: eth0 attempts 2

# Configure DAQ related options for inline operation. For more information, see READM
E.daq
#
# config daq: <type>
# config daq_dir: <dir>
# config daq_mode: <mode>
# config daq_var: <var>
#
# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
# <mode> ::= read-file | passive | inline
# <var> ::= arbitrary <name>=<value passed to DAQ
# <dir> ::= path as to where to look for DAQ module so's
```

Configure the Output



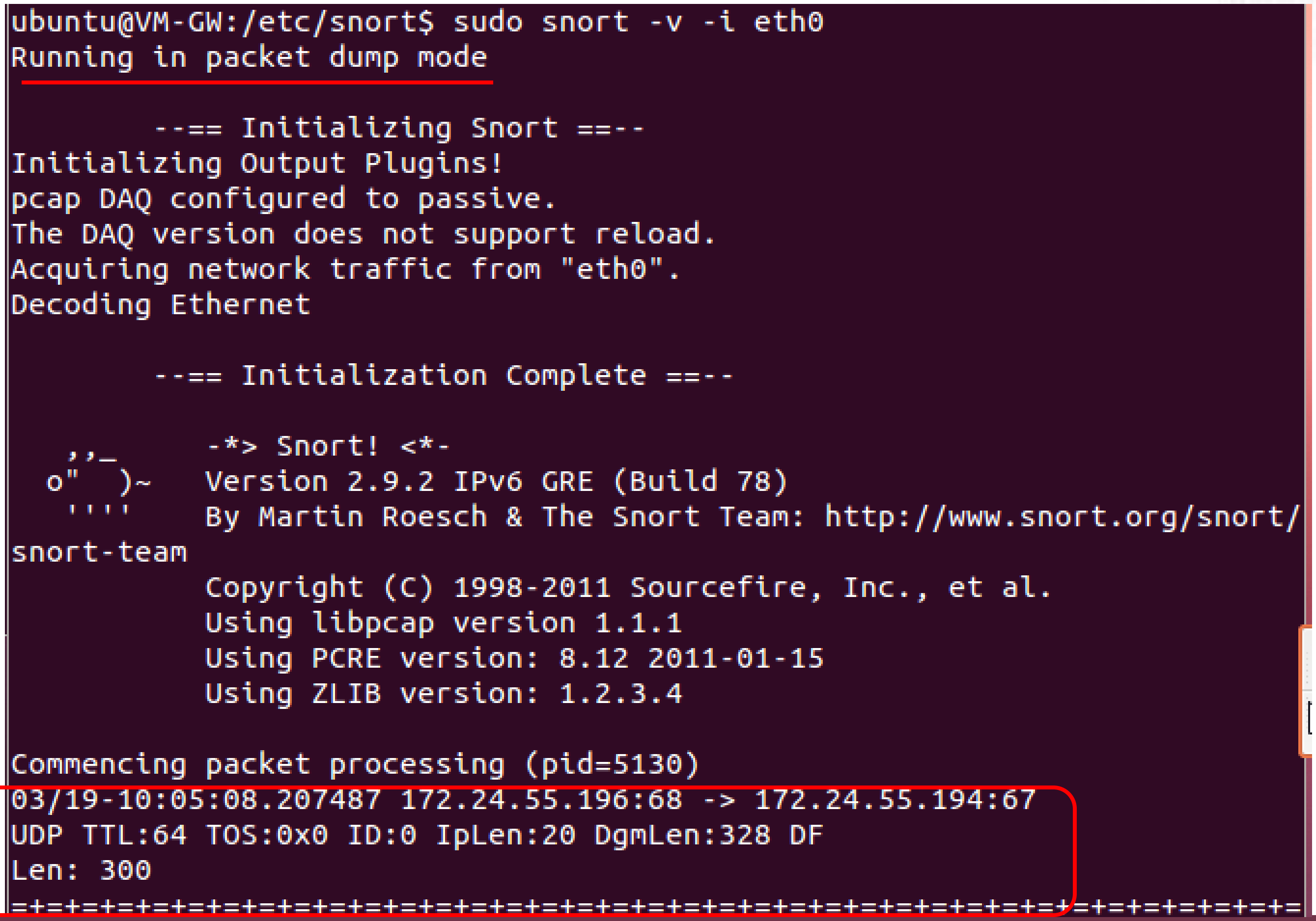
```
#####################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort - Output Modules
#####################################################

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_ev
ent_types

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
output log_tcpdump: tcpdump.log

# database
# output database: alert, <db_type>, user=<username> password=<password> test dbname=
<name> host=<hostname>
# output database: log, <db_type>, user=<username> password=<password> test dbname=<n
ame> host=<hostname>
```

Customize the Rule Set



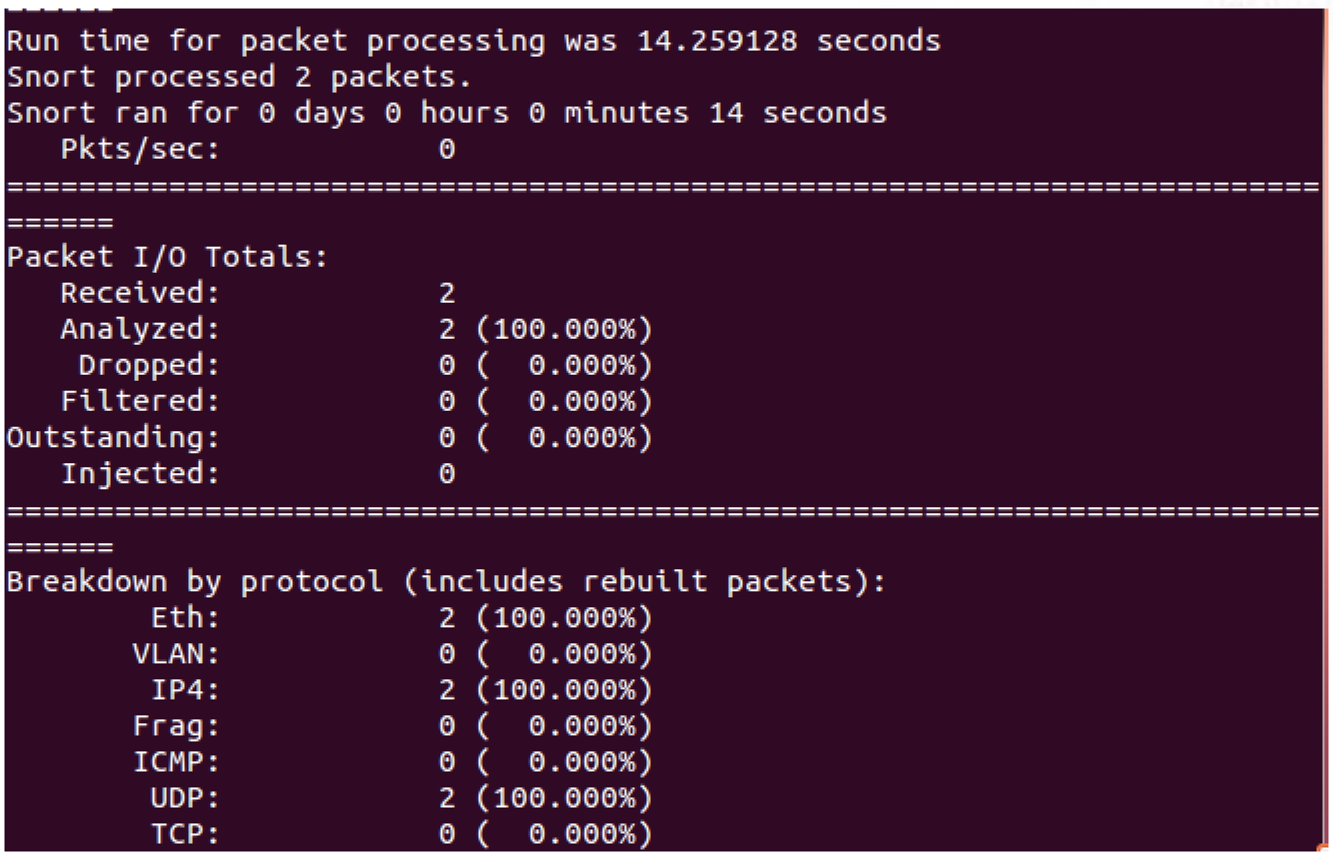
```
###################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
###################################################

# site specific rules
include $RULE_PATH/local.rules

include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
# include $RULE_PATH/blacklist.rules
# include $RULE_PATH/botnet-cnc.rules
include $RULE_PATH/chat.rules
# include $RULE_PATH/content-replace.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/community-dos.rules
include $RULE_PATH/exploit.rules
```

Running snort in a sniffer mode



```
ubuntu@VM-GW:/etc/snort$ sudo snort -v -i eth0
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.2 IPv6 GRE (Build 78)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/
snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.3.4

Commencing packet processing (pid=5130)
03/19-10:05:08.207487 172.24.55.196:68 -> 172.24.55.194:67
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:328 DF
Len: 300
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

Statistic Data



```
Run time for packet processing was 14.259128 seconds
Snort processed 2 packets.
Snort ran for 0 days 0 hours 0 minutes 14 seconds
   Pkts/sec:              0
===========================================================================
======
Packet I/O Totals:
   Received:              2
   Analyzed:              2 (100.000%)
    Dropped:              0 (   0.000%)
   Filtered:              0 (   0.000%)
Outstanding:              0 (   0.000%)
   Injected:              0
===========================================================================
======
Breakdown by protocol (includes rebuilt packets):
        Eth:              2 (100.000%)
       VLAN:              0 (   0.000%)
        IP4:              2 (100.000%)
       Frag:              0 (   0.000%)
       ICMP:              0 (   0.000%)
        UDP:              2 (100.000%)
        TCP:              0 (   0.000%)
```

Snort Modes

Using Snort

- Three main operational modes
  - sniffer mode: snort will read the network traffic and print them to the screen.
  - packet logger mode: snort will record the network traffic on a file
  - IDS mode: network traffic matching security rules will be recorded
  - IPS mode: also known as snort-inline (IPS = Intrusion prevention system)
- 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.

Using Snort – Sniffer Mode

- Works much like tcpdump, decodes packets and dumps them to stdout (packet dump mode).
  - Print out the **TCP/IP packet headers** to the screen (i.e. sniffer mode):

    ```
    > snort -i eth0 -v
    ```

    display the IP and TCP/UDP/ICMP headers on the console.
  - Check the **application data** in transit, try the following:

    ```
    > snort -i eth0 -vd
    ```

    display the packet data as well as the headers.
  - Show **the data link layer headers**, do this:

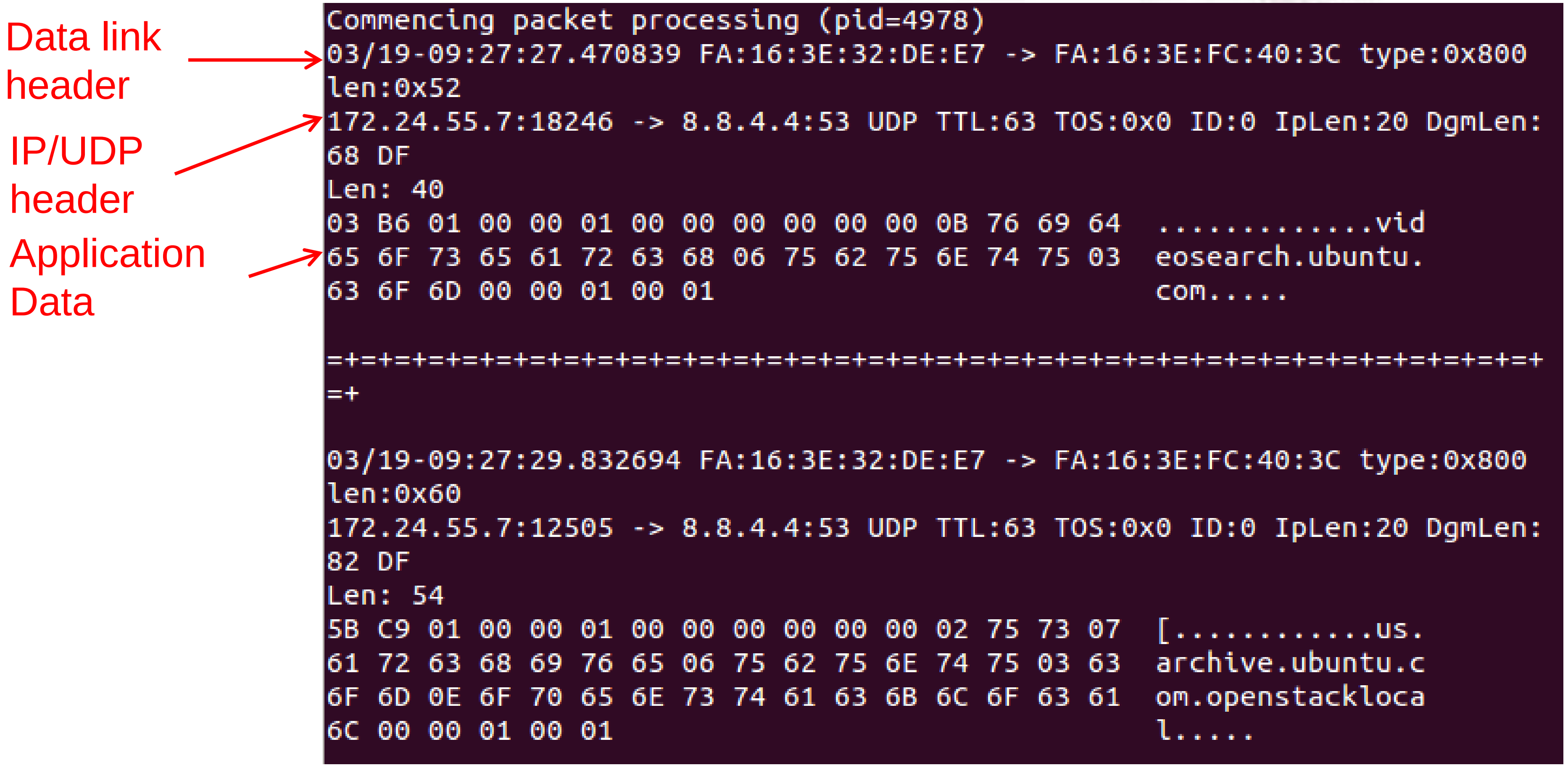
    ```
    > snort -i eth0 -vde
    ```

    As an aside, these switches may be divided up or smashed together in any combination. The last command could also be typed out as:

    ```
    > snort -i eth0 -d -v -e
    ```

    and it would do the same thing.

What Do The Packet Dumps Look Like?



```
Commencing packet processing (pid=4978)
03/19-09:27:27.470839 FA:16:3E:32:DE:E7 -> FA:16:3E:FC:40:3C type:0x800
len:0x52
172.24.55.7:18246 -> 8.8.4.4:53 UDP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:
68 DF
Len: 40
03 B6 01 00 00 01 00 00 00 00 00 00 0B 76 69 64  .............vid
65 6F 73 65 61 72 63 68 06 75 62 75 6E 74 75 03  eosearch.ubuntu.
63 6F 6D 00 00 01 00 01                          com.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+

03/19-09:27:29.832694 FA:16:3E:32:DE:E7 -> FA:16:3E:FC:40:3C type:0x800
len:0x60
172.24.55.7:12505 -> 8.8.4.4:53 UDP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:
82 DF
Len: 54
5B C9 01 00 00 01 00 00 00 00 00 00 02 75 73 07  [............us.
61 72 63 68 69 76 65 06 75 62 75 6E 74 75 03 63  archive.ubuntu.c
6F 6D 0E 6F 70 65 6E 73 74 61 63 6B 6C 6F 63 61  om.openstackloca
6C 00 00 01 00 01                                l.....
```

Packet Logger Mode

- Specify a logging directory to enable the packet logger mode:

  ```
  > snort –i eth0 -dev -l ./log
  ```

  – This assumes you have a directory named log in the current directory.

- Specify the log relative to the home network:

  ```
  > snort –i eth0 -dev -l ./log -h 192.168.1.0/24
  ```

  – Print out the data link and TCP/IP headers as well as application data into the directory ./log, and log the packets relative to the 192.168.1.0 class C network.

- Specify the log with a more compact form in binary mode. Binary mode logs the packets in tcpdump format to a single **binary** file in the logging directory:

  ```
  > snort –i eth0 -l ./log -b
  ```

- **Read a log file** in sniffer mode and dump the packets to the screen:

  ```
  > snort –i eth0 -dv -r packet.log
  ```

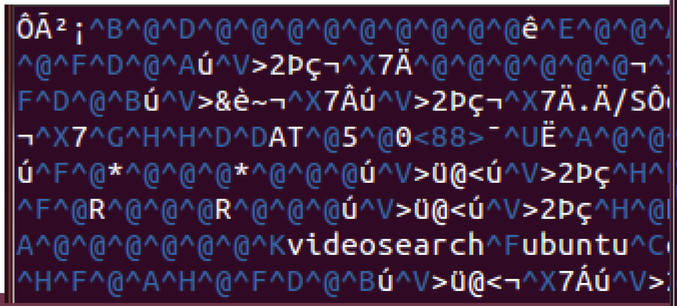
- Check **the ICMP packets only** in the log file:

  ```
  > snort –i eth0 -dvr packet.log icmp
  ```

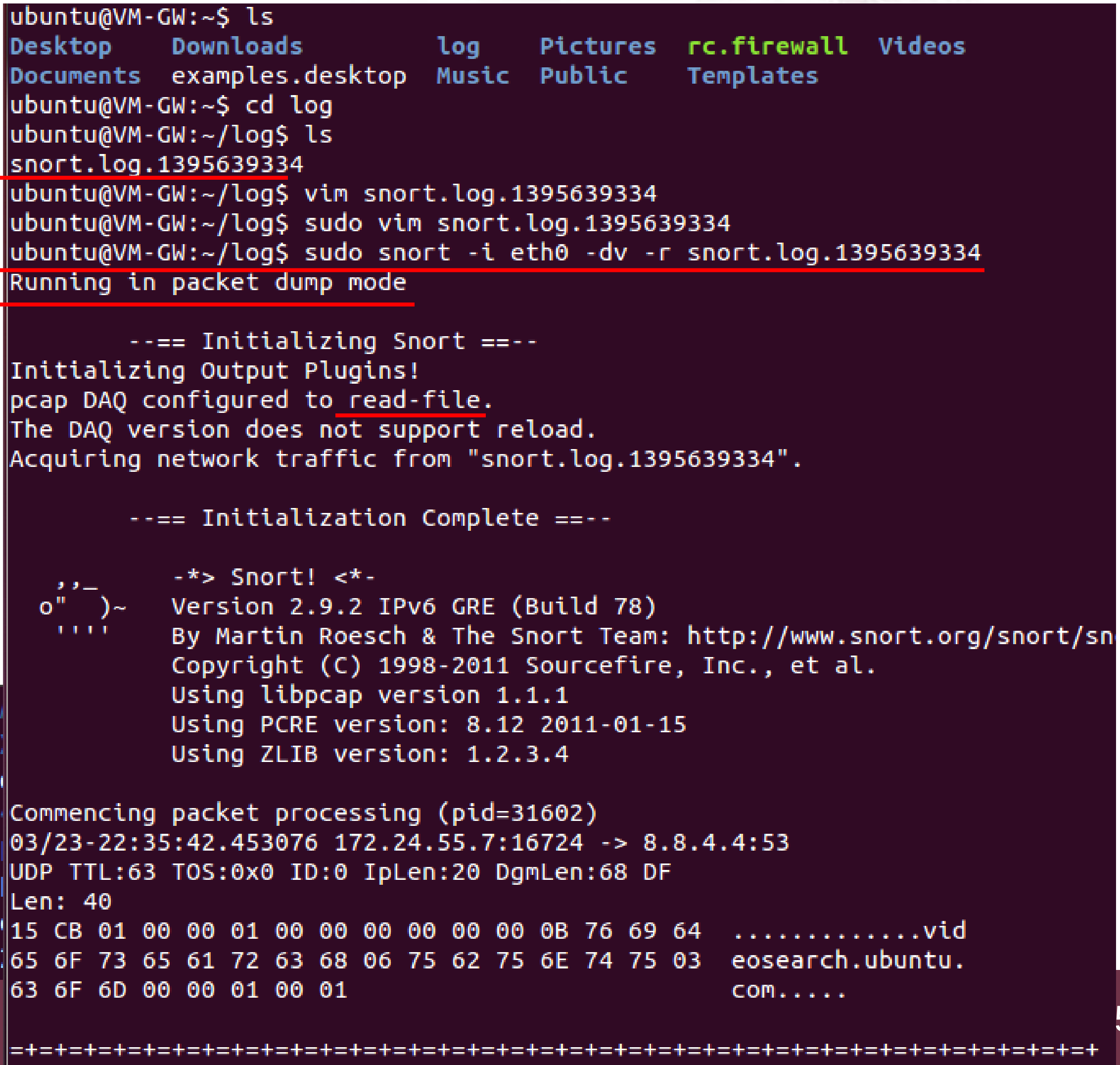
Read a log file with Snort

Log file

The contents of the Log file



```
ÔĀ²¡^B^@^D^@^@^@^@^@^@^@^@^@ê^E^@^@^
^@^F^D^@^Aú^V>2Ϸç¬^X7Ä^@^@^@^@^@^@¬^
F^D^@^Bú^V>&è~¬^X7Âú^V>2Ϸç¬^X7Ä.Ä/SÔ
¬^X7^G^H^H^D^DAT^@5^@0<88>¯^UË^A^@^@
ú^F^@*^@^@^@*^@^@^@ú^V>ü@<ú^V>2Ϸç^H^
^F^@R^@^@^@R^@^@^@ú^V>ü@<ú^V>2Ϸç^H^@
A^@^@^@^@^@^@^Kvideosearch^Fubuntu^C
^H^F^@^A^H^@^F^D^@^Bú^V>ü@<¬^X7Áú^V>
```



```
ubuntu@VM-GW:~$ ls
Desktop     Downloads          log     Pictures   rc.firewall  Videos
Documents   examples.desktop   Music   Public     Templates
ubuntu@VM-GW:~$ cd log
ubuntu@VM-GW:~/log$ ls
snort.log.1395639334
ubuntu@VM-GW:~/log$ vim snort.log.1395639334
ubuntu@VM-GW:~/log$ sudo vim snort.log.1395639334
ubuntu@VM-GW:~/log$ sudo snort -i eth0 -dv -r snort.log.1395639334
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to read-file.
The DAQ version does not support reload.
Acquiring network traffic from "snort.log.1395639334".

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.2 IPv6 GRE (Build 78)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/sn
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.1.1
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.3.4

Commencing packet processing (pid=31602)
03/23-22:35:42.453076 172.24.55.7:16724 -> 8.8.4.4:53
UDP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:68 DF
Len: 40
15 CB 01 00 00 01 00 00 00 00 00 00 0B 76 69 64  .............vid
65 6F 73 65 61 72 63 68 06 75 62 75 6E 74 75 03  eosearch.ubuntu.
63 6F 6D 00 00 01 00 01                          com.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

NIDS Mode

- Uses all phases of Snort + plug-ins to analyze traffic for both misuse detection and anomalous activity
  - To enable NIDS mode so that you don't record every single packet sent down the wire, try this:

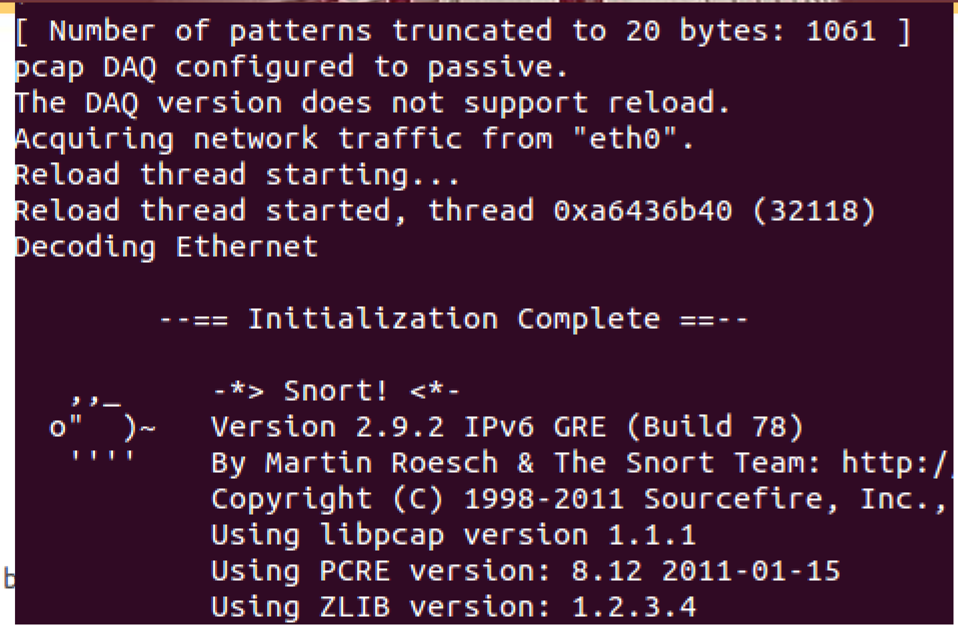
```
> snort -i eth0 -dev -l ./log -h 192.168.1.0/24 -c snort.conf
```

- snort.conf is the name of your rules file.
- If you don't specify an output directory, it will default to **/var/log/snort**.
- -v and -e switch can be removed to speed up the detection.

```
> snort -i eth0 -d -h 192.168.1.0/24 -l ./log -c snort.conf
```

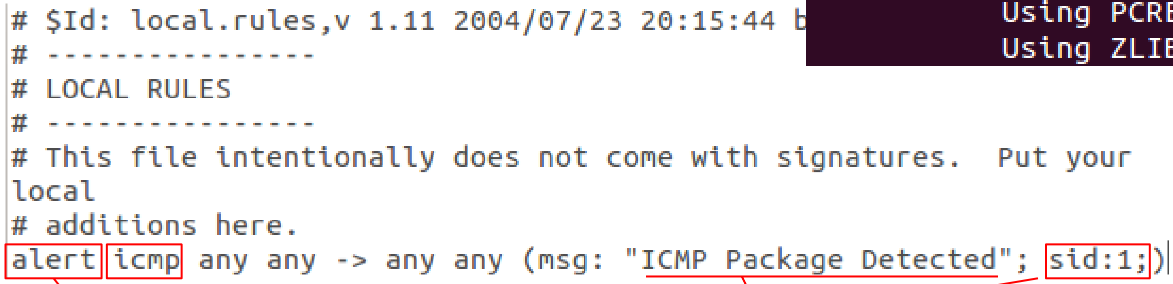
- This will configure Snort to run in its most basic NIDS form, logging packets that trigger rules specified in the snort.conf in plain ASCII to disk using a hierarchical directory structure (just like packet logger mode).

Detect ICMP packet with NIDS mode



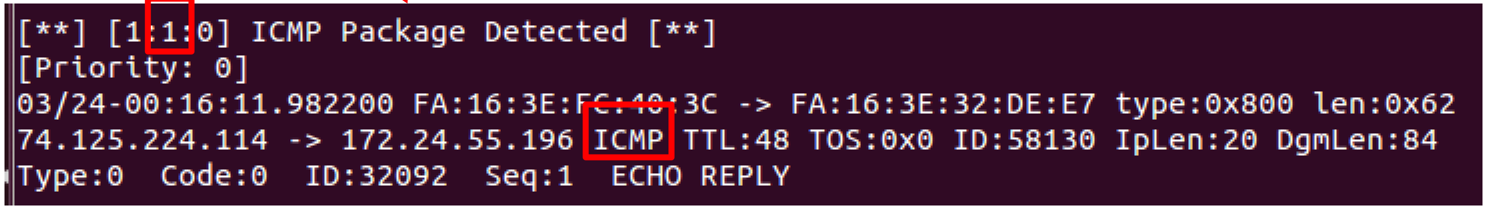
```
[ Number of patterns truncated to 20 bytes: 1061 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "eth0".
Reload thread starting...
Reload thread started, thread 0xa6436b40 (32118)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.2 IPv6 GRE (Build 78)
   ''''    By Martin Roesch & The Snort Team: http:/
           Copyright (C) 1998-2011 Sourcefire, Inc.,
           Using libpcap version 1.1.1
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.3.4
```

Local.rules



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 b
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your
local
# additions here.
alert icmp any any -> any any (msg: "ICMP Package Detected"; sid:1;)
```

alert



```
[**] [1:1:0] ICMP Package Detected [**]
[Priority: 0]
03/24-00:16:11.982200 FA:16:3E:FC:40:3C -> FA:16:3E:32:DE:E7 type:0x800 len:0x62
74.125.224.114 -> 172.24.55.196 ICMP TTL:48 TOS:0x0 ID:58130 IpLen:20 DgmLen:84
Type:0  Code:0  ID:32092  Seq:1  ECHO REPLY
```

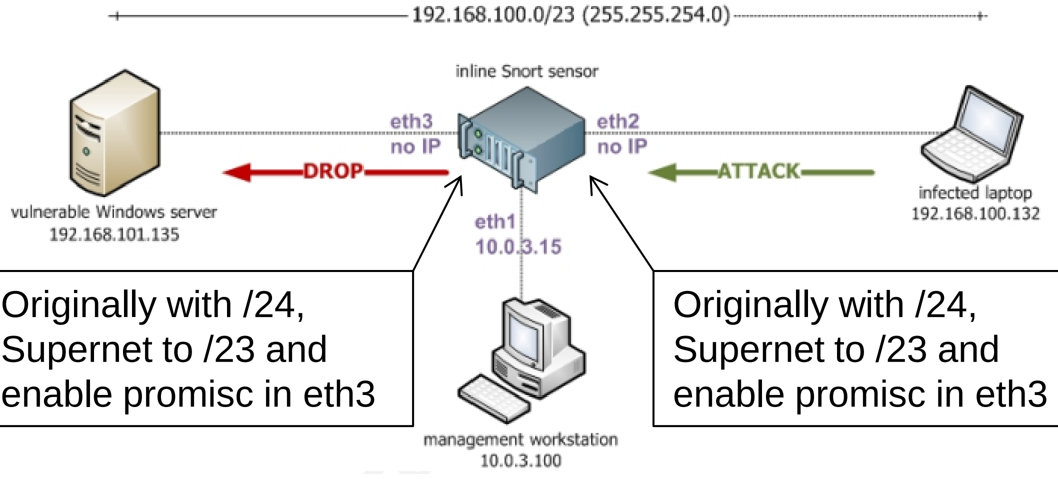
IPS Mode (Inline Mode)

- Snort traditionally only rises alerts and logs traffic.
- In IPS mode snort is able to drop packets
- The network flow must go through snort (inline mode)
- Requirements:
  - Put Snort sensor on the wire and make traffic go through it.
    - **-i eth1:eth2**
  - Specify command line options
    - **-Q** `or` **--daq-mode inline**
  - Specify DAQ through command line or DAQ configuration
    - **--daq <type>**
  - DAQ configuration in snort.conf

```
config daq: <type>              <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
config daq_dir: <dir>           <mode> ::= read-file | passive | inline
config daq_var: <var>           <var> ::= arbitrary <name>=<value> passed to DAQ
config daq_mode: <mode>         <dir> ::= path where to look for DAQ module so's
```

DAQ afpacket

- afpacket module is available only on Linux.
- It leverages Snort rules and two net interfaces to drop suspicious traffic without having to rely on a separate, external firewall.
- Instead, Snort configures a net interface pair as a transparent bridge to forward traffic so both network segments appear to be part of one broadcast domain.



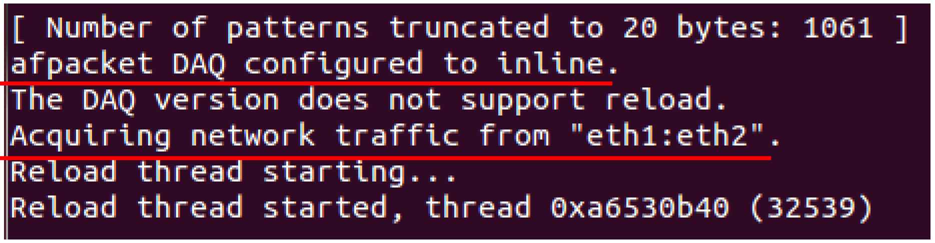
Originally with /24, Supernet to /23 and enable promisc in eth3

Originally with /24, Supernet to /23 and enable promisc in eth3

Run Snort with inline afpacket

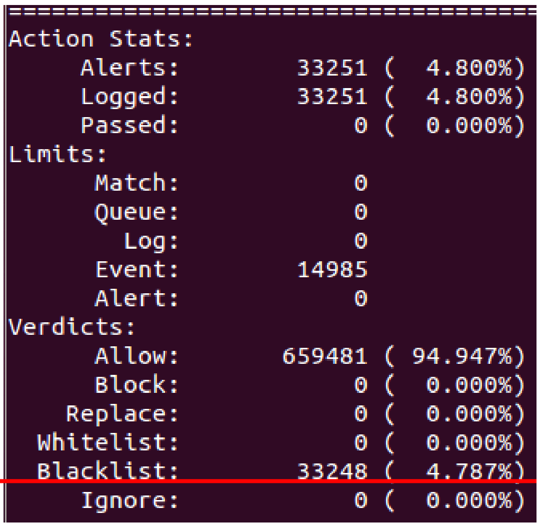
> Sudo snort –Q –daq afpacket –i eth1:eth2 –l ./log –c /etc/snort/snort.conf

with rule → **drop** icmp any any -> any any (msg: “ICMP Package Detected”; sid:1;)

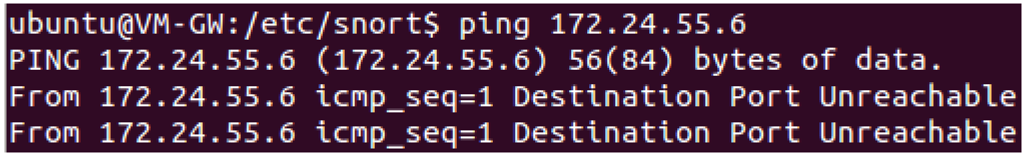


```
[ Number of patterns truncated to 20 bytes: 1061 ]
afpacket DAQ configured to inline.
The DAQ version does not support reload.
Acquiring network traffic from "eth1:eth2".
Reload thread starting...
Reload thread started, thread 0xa6530b40 (32539)
```

drop packets

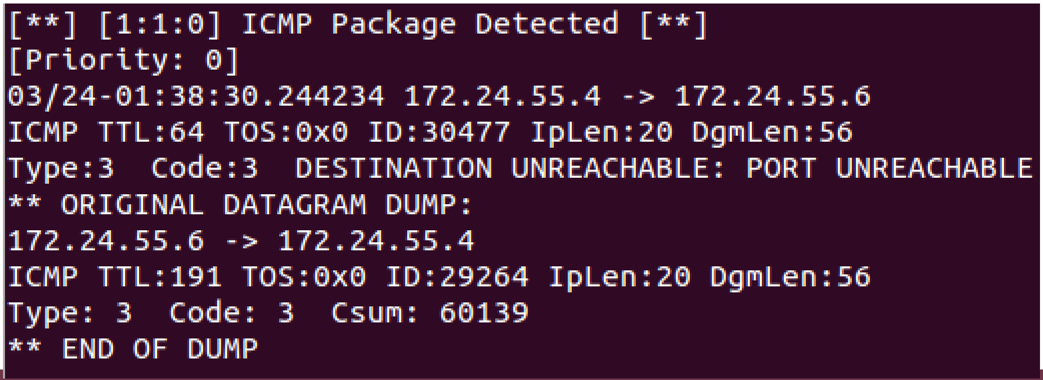


```
====================================
Action Stats:
     Alerts:        33251 (  4.800%)
     Logged:        33251 (  4.800%)
     Passed:            0 (  0.000%)
Limits:
      Match:            0
      Queue:            0
        Log:            0
      Event:        14985
      Alert:            0
Verdicts:
      Allow:       659481 ( 94.947%)
      Block:            0 (  0.000%)
    Replace:            0 (  0.000%)
  Whitelist:            0 (  0.000%)
  Blacklist:        33248 (  4.787%)
     Ignore:            0 (  0.000%)
```



```
ubuntu@VM-GW:/etc/snort$ ping 172.24.55.6
PING 172.24.55.6 (172.24.55.6) 56(84) bytes of data.
From 172.24.55.6 icmp_seq=1 Destination Port Unreachable
From 172.24.55.6 icmp_seq=1 Destination Port Unreachable
```

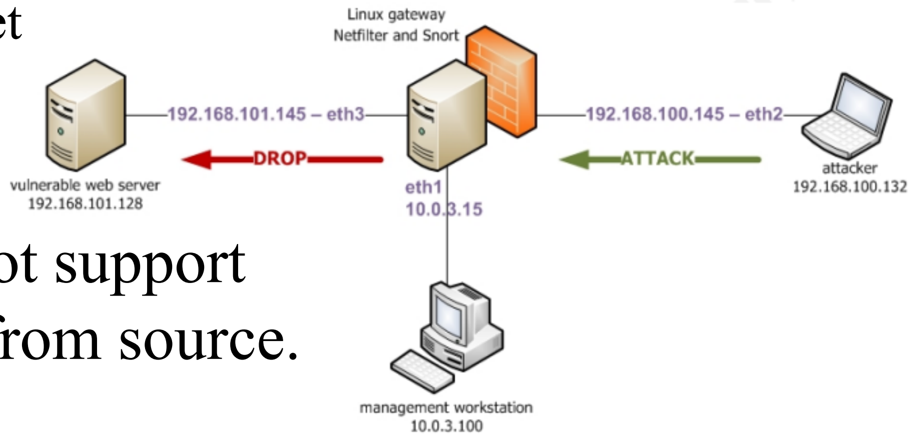
alert



```
[**] [1:1:0] ICMP Package Detected [**]
[Priority: 0]
03/24-01:38:30.244234 172.24.55.4 -> 172.24.55.6
ICMP TTL:64 TOS:0x0 ID:30477 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
172.24.55.6 -> 172.24.55.4
ICMP TTL:191 TOS:0x0 ID:29264 IpLen:20 DgmLen:56
Type: 3  Code: 3  Csum: 60139
** END OF DUMP
```

DAQ NFQ

- The nfqueue module leverages the NFQUEUE target in iptables to move packets from the kernel to a user space application for evaluation.
  - iptables –A FORWARD –j NFQUEUE
- Requirements:
  - Enable IP forwarding in the kernel
  - Configure Snort rules to drop the packet
  - Configure iptables to filter the traffic
  - Configure iptables to queue traffic
- The default DAQ (daq0.6) does not support NFQ, you need to build a new daq from source.



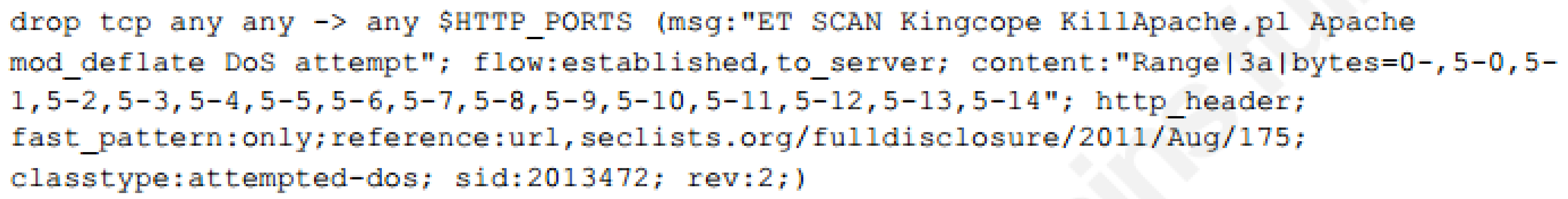
DAQ NFQ Example

> snort –daq nfq --daq-ver queue=2 –Q –l ./log –c /etc/snort/snort.conf

with iptables rule:

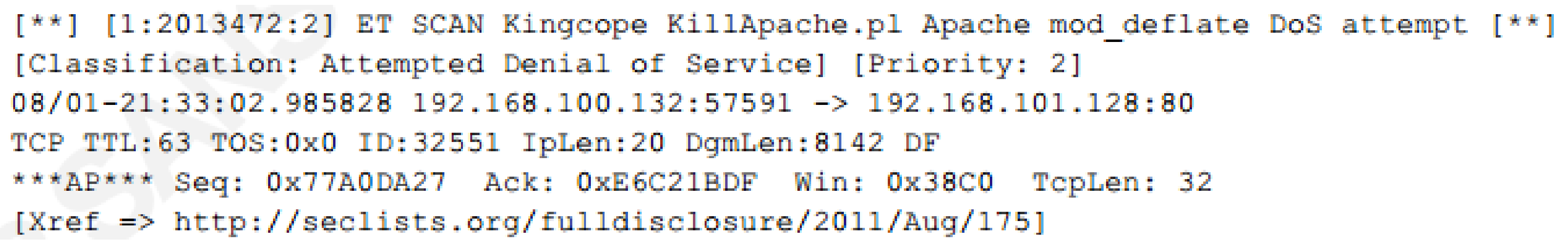
> iptables –A INPUT –p tcp --dport 80 –j NFQUEUE –queue-num 2

and Snort rule:



```
drop tcp any any -> any $HTTP_PORTS (msg:"ET SCAN Kingcope KillApache.pl Apache
mod_deflate DoS attempt"; flow:established,to_server; content:"Range|3a|bytes=0-,5-0,5-
1,5-2,5-3,5-4,5-5,5-6,5-7,5-8,5-9,5-10,5-11,5-12,5-13,5-14"; http_header;
fast_pattern:only;reference:url,seclists.org/fulldisclosure/2011/Aug/175;
classtype:attempted-dos; sid:2013472; rev:2;)
```

- An exploit written by Kingcope (2011) to trigger a DoS vulnerability on an unpatched Apache web server.



```
[**] [1:2013472:2] ET SCAN Kingcope KillApache.pl Apache mod_deflate DoS attempt [**]
[Classification: Attempted Denial of Service] [Priority: 2]
08/01-21:33:02.985828 192.168.100.132:57591 -> 192.168.101.128:80
TCP TTL:63 TOS:0x0 ID:32551 IpLen:20 DgmLen:8142 DF
***AP*** Seq: 0x77A0DA27  Ack: 0xE6C21BDF  Win: 0x38C0  TcpLen: 32
[Xref => http://seclists.org/fulldisclosure/2011/Aug/175]
```

Snort IPS action

- **drop**: Tell iptables to drop the packet and log it via usual Snort means
- **reject**: Tell iptables to drop the packet, log it via usual Snort means, and send a TCP reset if the protocol is TCP or an icmp port unreachable if the protocol is UDP
- **sdrop**: The sdrop rule type will tell iptables to drop the packet. Nothing is logged

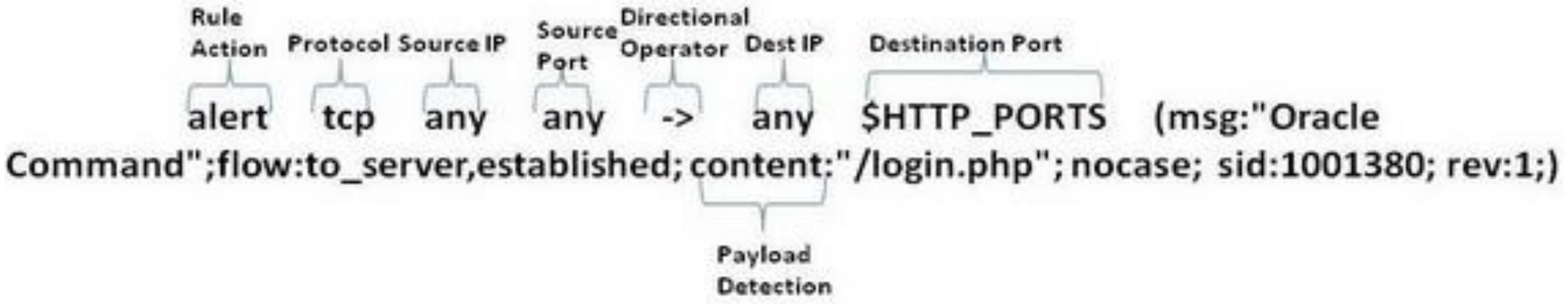
Snort Rules Illustration

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 27374 -> $HOME_NET any (msg:"BACKDOOR subseven
   22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
   reference:arachnids,485; reference:url,www.hackfix.org/subseven/;
   sid:103; rev:4;)
```

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



```
alert tcp any any -> any $HTTP_PORTS (msg:"Oracle
Command";flow:to_server,established;content:"/login.php";nocase; sid:1001380; rev:1;)
```

Snort Rules

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
   subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
   reference:arachnids,485;
   reference:url,www.hackfix.org/subseven/; sid:103; rev:4;)
```

- `alert` action to take; also `log, pass, activate,` **and drop in inline mode**
- `tcp` protocol; also `udp, icmp, ip`
- `$EXTERNAL_NET` source address; this is a variable – specific IP is ok
- `27374` source port; also `any`, negation (`!21`), range (`1:1024`)
- `->` direction; direction; best not to change this, although <> is allowed
  `$HOME_NET` destination address; this is also a variable here
- `any` destination port

Snort Rules

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
   subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
   reference:arachnids,485;
   reference:url,www.hackfix.org/subseven/; sid:103; rev:4;)
```

- `msg:"BACKDOOR subseven 22";` message to appear in logs
- `flags: A+;` tcp flags; many options, like `SA, SA+, !R, SF* (what do those flag mean ? Explore that by yourself)`
- `content: "|0d0…0a|";` binary data to check in packet; content without | (pipe) characters do simple content matches
- `reference…;` where to go to look for background on this rule
- `sid:103;` rule identifier
- `rev:4;` rule revision number
- other rule options possible, like `offset, depth, nocase`
  - **depth: how far into a packet Snort should check**
  - **nocase: case insensitive**
  - **offset: how far in a packet Snort start to check**

Snort Rules

Under /etc/snort/rules

• bad-traffic.rules	exploit.rules	scan.rules
• finger.rules	ftp.rules	telnet.rules
• smtp.rules	rpc.rules	rservices.rules
• dos.rules	ddos.rules	dns.rules
• tftp.rules	web-cgi.rules	web-coldfusion.rules
• web-frontpage.rules	web-iis.rules	web-misc.rules
• web-attacks.rules	sql.rules	x11.rules
• icmp.rules	netbios.rules	misc.rules
• backdoor.rules	shellcode.rules	policy.rules
• porn.rules	info.rules	icmp-info.rules
• virus.rules	local.rules	attack-responses.rules

Snort Rules

- Rules which actually caught intrusions
  - caught compromise of Microsoft SQL Server

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 1433 (msg:"MS-SQL
xp_cmdshell - program execution"; content:
"x|00|p|00|_|00|c|00|m|00|d|00|s|00|h|00|e|00|l|00|l|00|";
nocase; flags:A+; classtype:attempted-user; sid:687; rev:3;)
```

  - caught Code Red infection

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS
cmd.exe access"; flags: A+; content:"cmd.exe"; nocase;
classtype:web-application-attack; sid:1002; rev:2;)
```

  - caught anonymous ftp server

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"INFO FTP
\"MKD / \" possible warez site"; flags: A+; content:"MKD / ";
nocase; depth: 6; classtype:misc-activity; sid:554; rev:3;)
```

Output and logging Modules

Alert output Module

Output modules are loaded at runtime by specifying the output keyword in the config file: **output <name>: <options>**, or specify switch "-A" in the command line.

- **alert_fast**:
  - print alerts in a quick one-line format to a specified output file.
  - -A fast (in command line)
  - output alert_fast: filename (the default name is <logdir>/alert)
- **alert_full**:
  - print alert messages with full packet headers (slows Snort down considerably)
  - Inside the logging directory, a directory will be created per IP.
  - output alert_full: filename

Alert output Module (cont.)

- alert_unixsock
  - Sets up a UNIX domain socket and sends alert reports to it. External programs/processes can listen in on this socket and receive Snort alert and packet data in real time.
  - -A unsock
  - output alert_unixsock
- log_tcpdump
  - logs packets to a tcpdump-formatted file, useful for performing post-process analysis.
  - output log_tcpdump filename

Alert output Module (cont.)

- alert_syslog
  - sends alerts to the syslog facility, allows the user to specify the logging facility and priority within the Snort config file, giving users greater flexibility in logging alerts.
  - Format: output alert_syslog: <facility> <priority> <options>
  - E.g.: output alert_syslog: host=10.1.1.1:514, LOG_AUTH LOG_ALERT

Facilities

- log_auth
- log_authpriv
- log_daemon
- log_local0
- log_local1
- log_local2
- log_local3
- log_local4
- log_local5
- log_local6
- log_local7
- log_user

Priorities

- log_emerg
- log_alert
- log_crit
- log_err
- log_warning
- log_notice
- log_info
- log_debug

Optoins

- log_cons
- log_ndelay
- log_perror
- log_pid

New output format unified2

- Unified2 can work in one of three modes, packet logging, alert logging, or true unified logging.
  - *Packet logging* includes a capture of the entire packet and is specified with **log_unified2**.
  - *Alert logging* will only log events and is specified with **alert_unified2**.
  - To include both logging styles in a single, unified file, simply specify **unified2**.
  - Examles:

```
output alert_unified2: filename snort.alert, limit 128, nostamp
output log_unified2: filename snort.log, limit 128, nostamp
output unified2: filename merged.log, limit 128, nostamp
```

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