



Intrusion Detection Systems & Snort



IDS – Intrusion Detection Systems

- Intrusion

- Unwanted and unauthorized intentional access of computerized network resources

- IDS Examples

- Car Alarms, Burglar Alarms, Motion Sensors

- Helps detect intrusion and intrusion attempts in a network

- Software based or combination of hardware and software in a stand-alone device



IDS Types

- Network-Based (NIDS)
 - Monitor network links and backbones
- Host-Based (HIDS)
 - Operate on hosts and defends and monitor the operating and file systems for signs of intrusion
- Distributed (DIDS)
 - Groups of IDSes working as remote sensors that report back to a central management station
- Gateway
 - Network IDS deployed at a network gateway to monitor traffic passing in and out
- Application
 - Follow flow of application layer protocols



Network IDS

- Monitors an entire network segment
- Should be connected to span port or network tap
- NIC operates in promiscuous mode
 - Allows it to view traffic not sent to its MAC address
 - CPU intensive for slow hosts
- Must be careful because of wiretap laws



Host-Based IDS

- Protects only the host system on which it resides
- NIC operates in non-promiscuous mode
- Sees system calls, file system modifications, and system logs
- Can tailor rules to match specific needs of host
 - Ex: Don't use DNS rules for machine not hosting DNS
 - Ex: Mail server HIDS has special rules to detect mail server exploits
 - Reduces false-positives and processor overhead



Distributed IDS

- Made up of NIDS, HIDS, or both
- Sensors are located across the network and report to a centralized management station (CMS)
- CMS stores attack signature database and sends them to sensors as needed
- Sensors can have specialized rule sets (don't have to all be the same)
- Use a VPN or encryption for communication between sensors and CMS for extra protection



IDS Detection Types

■ Signature Detection

- Like antivirus, uses database of traffic or activity patterns related to known attacks
- Most widely used

■ Anomaly Detection

- Uses rules or predefined concepts about normal and abnormal activity (aka heuristics)
- Can be constructed using statistical sampling, rule-based approaches, or neural networks



IDS Response Types

- Passive Response

- Generate alerts or log entries in response to alerts

- Active Response

- Interferes or manipulates network traffic in response to alerts

- Examples

- Send reset packets to disrupt TCP connections
- Add attacking machine to block lists
- Drop packets



IDS Problems

- **False Positive**

- Alert that triggers on normal traffic where no intrusion or attack is underway
- Create tremendous amounts of data in new IDSes
- Reduce through tuning over time

- **False Negative**

- Failure of a rule to trigger when an actual attack is underway

- **Fragmentation and Overlapping**

- Makes packets reassemble differently than they appear to

- **Encryption**

- Removes capability of IDS to alert on packet payloads reliably





What is Snort?

- Open source signature based NIDS
- Performs real-time traffic analysis and packet logging on IP networks
- Also performs protocol analysis
 - Ex: TCP, UDP, ICMP
- Small footprint and small system requirements
- Runs on a variety of operating systems
 - Linux, Solaris, Mac OS X, HP-UX, IRIX, Windows



Snort History

- Created by Marty Roesch
- December 1998 - download via Packet Storm
 - 1,600 lines of code and 2 files
 - Packet sniffing only
- January 1999 – signature-based analysis feature added
- Roesch started Sourcefire and hired people to work on Snort full-time
- Now on version 2.7.0.1



Hardware Requirements

- Main Goal: No Packet Loss
- Reasonably fast hard drive
 - 60 GB minimum
- NIC as fast/faster than the rest of network
 - 100 Mbps minimum, get gigabit card if possible
- Recommend second NIC for remote access and administrative communication
- 2-4 GB RAM recommended
- Almost any OS, but Linux preferred



Installing Snort

- Download latest release from
 - <http://www.snort.org/dl/>
- Install documentation available at
 - <http://www.snort.org/docs/>
- Different install options based on desired functionality



Installing Snort

■ Prerequisites

- Autoconf and automake*
- Gcc
- Lex and yacc
- Libcap from tcpdump.org
- Perl



Location Points

■ Network Tap

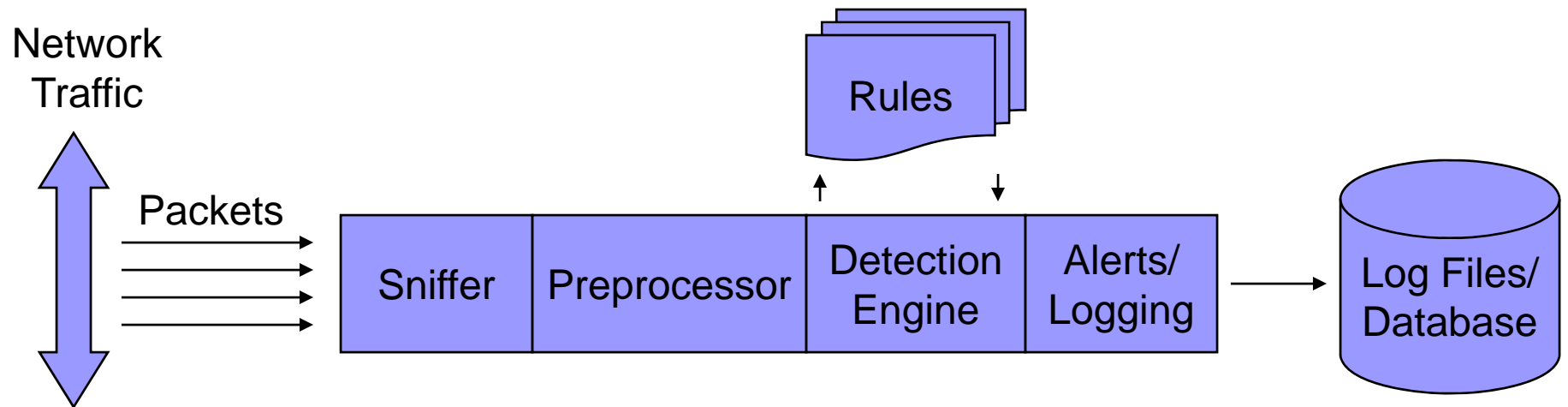
- Gives Snort permanent access for passive monitoring
- Create an access port used for collecting all data traveling across the network
- Uses regeneration of a full-duplex network signal

■ Span Ports

- Span ports on switches allow Snort to sniff multiple networks
- Do not give access to all network traffic
- Have to be greater than the amount of traffic
 - Ex: 3 100-Mbps ports requires a 300-Mbps+ span port

Snort Architecture

- Sniffer
- Preprocessor
- Detection Engine
- Alerts/Logging

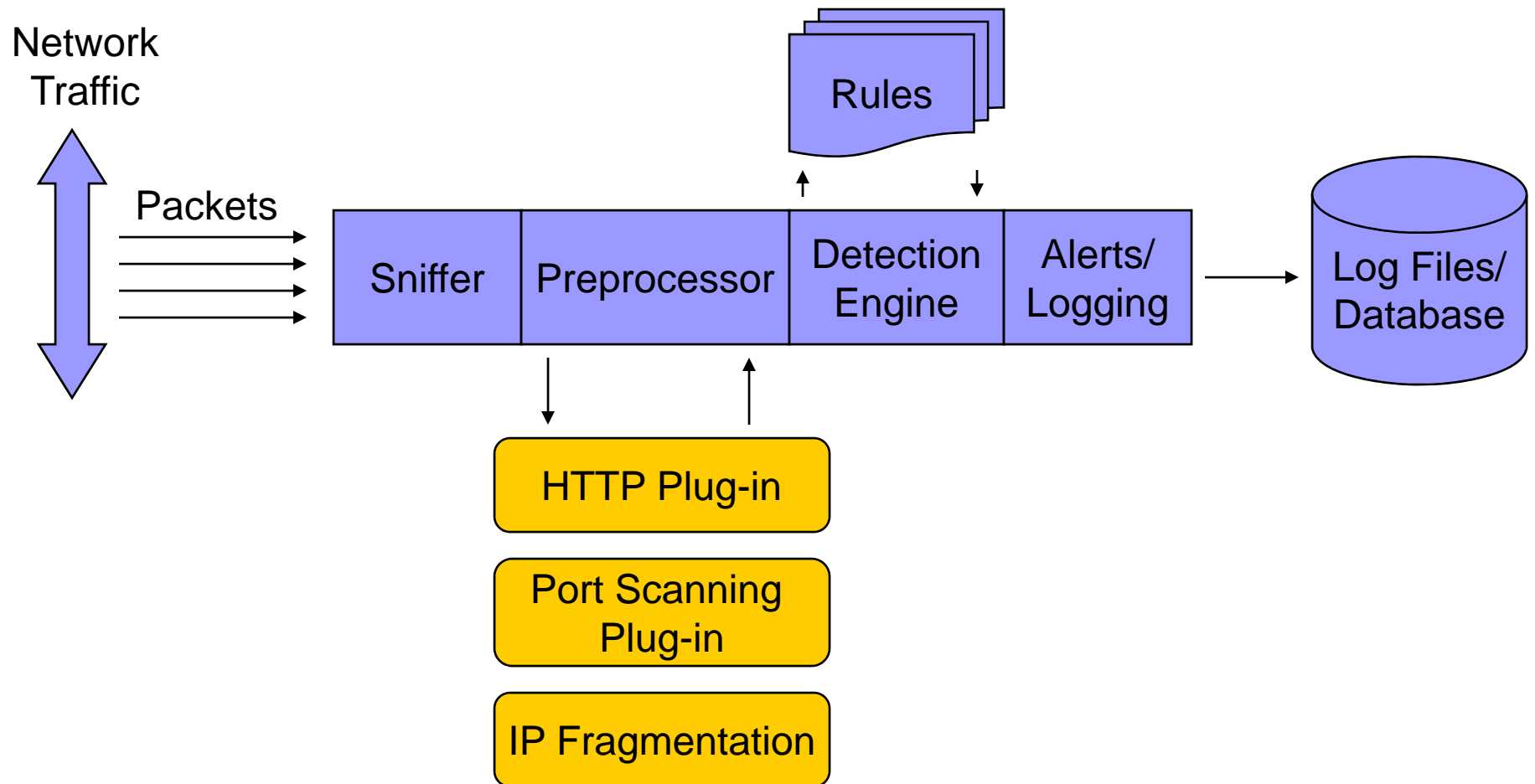




Preprocessor

- Takes raw packets and checks them against enabled plug-ins
 - Ex: HTTP plug-in, port scanner plug-in, etc
- Plug-ins check for certain types of behaviors in a packet
- Once a behavior is detected, the packet is sent to the detection engine
- Users can enable and disable plug-ins as needed
- Plug-ins remove unneeded functionality and increase efficiency

Preprocessor Flow





Preprocessor Plug-ins

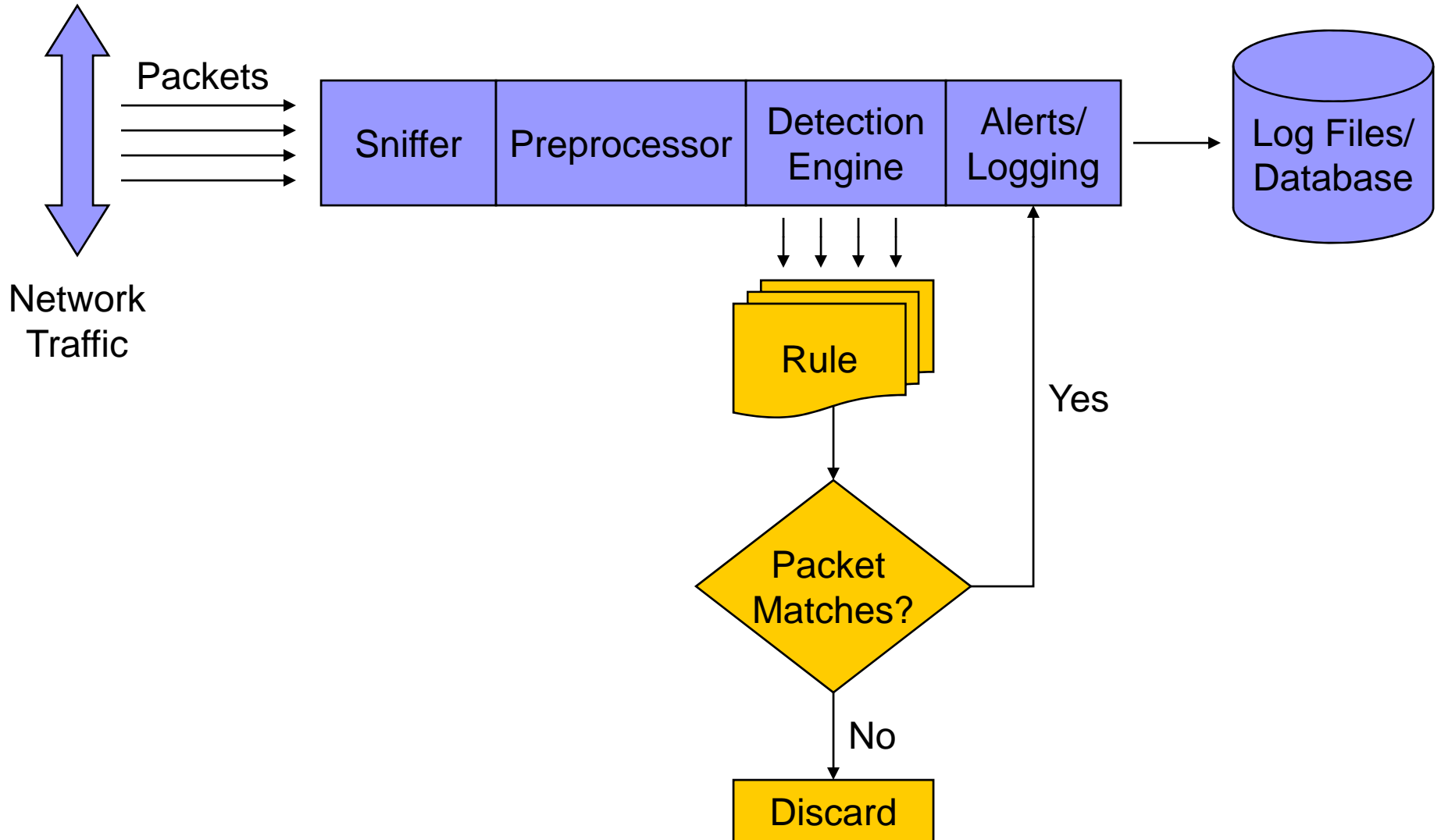
- frag2 & frag3
- flow
- stream4 & stream5
- telnet_decode
- HTTP Inspect
- rpc_Decode
- sfPortscan
- Back Orifice
- FTP_Telnet
- SMTP



Detection Engine

- Receives packets after preprocessors are through
- Takes data from preprocessors and checks it through the rule set
- If a rule matches, the packet is sent to the alert processor

Detection Engine Flow





Alerts/Logging

- If data matches a rule in detection engine, an alert is triggered
- Alert Options
 - Log file
 - Network connection
 - UNIX sockets
 - Windows SMB
 - SNMP Traps
 - Stored in Database



Alerts/Logging Plug-ins

- Perl
- PHP
- MySQL Database
- 3rd Party tools
 - SnortSnarf – HTML output
 - Snortplot.php – Perl script that plots attacks
 - Swatch – Real-time syslog monitor
 - ACID – Greater logging analysis



Snort Rules/Signatures

Purpose of Rules

- Describe a state of the network
- List an action to perform if the state is true

Example

If a packet containing “virus.exe” enters the network from an external IP and is destined for the HTTP server on port 80, send an alert.



Rule Example

```
alert tcp $EXTERNAL_NET any  
-> $HTTP_SERVER 80
```

```
(msg:"It's a Virus!"; flow:to_server;  
content:"virus.exe"; nocase;  
classtype:virus-attack; sid:100097;  
rev:1;)
```



Rule Anatomy

■ Part 1 - Rule Header

- Action – alert, pass, drop, reject, etc.
- Protocol – TCP, UDP, ICMP, or IP
 - (IP includes the TCP, UDP, and ICMP)
 - Other protocols can be specified using the rule proto option
- IPs – Individual IP, range of IPs, list of IPs, or “any”
- Ports – Individual port, range of ports, or “any”

Example Rule

```
alert tcp $EXTERNAL_NET any -> HTTP_SERVER 80
```

```
(msg:“It’s a Virus!”; flow:to_server; content:”virus.exe”; nocase;  
 classtype:virus-attack; sid:100097; rev:1;)
```



Rule Anatomy

■ Part 2 - Rule Options

- Rule Title – message describing event
- Flow – to_server, from_server, to_client, from_client, established, and stateless
 - Can eliminate half or more of traffic from pattern matching
- Content – content and uricontent
 - Content – match payload of packet
 - Uricontent – match normalized URL data (more efficient)

Example Rule

```
alert tcp $EXTERNAL_NET any -> HTTP_SERVER 80
```

```
(msg: "It's a Virus!"; flow:to_server; content:"virus.exe"; nocase;  
 classtype:virus-attack; sid:100097; rev:1;)
```



Rule Anatomy

■ More Rule Options

- Depth – specify where in packet we want to specify a match
 - Ex: content:"GET"; depth:10;
 - Match only if GET is within first 10 bytes of packet
- Offset – Ignore first x bytes and look from there until the end of the packet
 - Ex: content:"cmd.exe"; offset: 50;
- Combination
 - Ex: content:"target_string"; offset:100; depth:50;
- Within, distance, rawbytes



Rule Anatomy

■ Part 3 - Metadata

- Reference – url, bugtraq #, cve #, nessus #, and mcafee #
 - Allow quick reference for purpose of rule
- Classtype – web-application attack, misc-activity, unsuccessful-user, etc
 - Defined in classification.config
- SID – Snort's unique identifier for a rule
 - 1,000,001 – 1,999,999: reserved for local use
 - Other ranges are reserved for rule repositories
- Rev – Revision number
 - Like the SID, not required but highly recommended

Example Rule

```
alert tcp $EXTERNAL_NET any -> HTTP_SERVER 80
```

```
(msg: "It's a Virus!"; flow:to_server; content:"virus.exe"; nocase;  
classtype:virus-attack; sid:100097; rev:1;)
```



Existing Snort Rules

- Currently 9,500 existing rules
- Snort Repositories
 - Snort.org
 - Original repository, closed to development
 - Free
 - VRT (Vulnerability Response Team)
 - Updated by Sourcefire researches
 - Paid subscription
 - Bleeding Edge Threats
 - Under Berkeley Software Distribution License
 - Snort Community
 - Continuation of development of original repository
 - Free



Examples of Current Rules

- Metasploit signatures
- SolarWinds scanner's ICMP packets
- Buffer Overflows
 - Red Hat 1prd overflow, Linux samba overflow, IMAP login overflow, Linux mountd overflow
- Many viruses, worms, trojans, etc
- ORACLE drop table attempt
- MYSQL show databases attempt



The Snort Configuration File

- Define network variables
 - Ex: `var HOME_NET 10.1.1.0/24`
- Configure dynamic loaded libraries
- Enable and configure preprocessor plug-ins
 - Ex: `preprocessor stream4: enforce_state, keepstats`
- Enable and configure output plug-ins
 - Ex: `output log_tcpdump: tcpdump.log`
- Configure snort with config statements
- Customize rule set
 - Ex: `alert tcp any any -> any any (msg:"exe alert"; content:"MZP"; nocase; sid:100000000000; rev:1;)`



The Snort Configuration File

```
#-----  
# http://www.snort.org Snort 2.6.1.5 Ruleset  
# Contact: snort-sigs@lists.sourceforge.net  
#-----  
# $Id$  
#  
#####  
# This file contains a sample snort configuration.  
# You can take the following steps to create your own custom configuration:  
#  
# 1) Set the variables for your network  
# 2) Configure dynamic loaded libraries  
# 3) Configure preprocessors  
# 4) Configure output plugins  
# 5) Add any runtime config directives  
# 6) Customize your rule set  
#  
#####
```



Snort Modes

- Packet Sniffer
 - Reads packets off network and displays them on console
- Packet Logger
 - Logs packets to disk
- Network Intrusion Detection
 - Analyzes network traffic for matches against rules
 - Performs predefined actions upon alerts
- Inline Active Defense
 - Can drop packets when alerts are triggered



Sniffer Mode

- Displays data to console in packet dump format
- Snort -d -e -v
 - -v Put Snort in sniffing mode (TCP headers only)
 - -d Include all network layer headers (TCP, UDP, ICMP)
 - -e Include the data link layer headers



Example of Packet Dump Format

{date}-{time} {source-hw-address} -> {dest-hw-address} {type}
{length} {source-ip-address:port} -> {dest-ip-address:port}
{protocol} {TTL} {TOS} {ID}
{IP-length}
{datagram-length}
{payload-length}
{hex-dump} {ASCII-dump}



Packet Logging Mode

- Snort `-dev -l {logging-directory}`
 - Ex: `snort -dev -l /var/adm/snort/logs`
 - Logs packets in TCPDump format
- Binary Logging
 - Snort `-b -L {log-file}`
 - Fastest logging available
 - Can analyze data later in TCPDump or Ethereal



Snort in NIDS mode

- `Snort -dev -l /var/adm/snort/logs -c /root/mysnort.conf`
 - `-c` Specifies the configuration file to be used
 - User can enable different preprocessor and output plug-ins
 - Define custom rules in the configuration file



Stopping Snort

- To stop snort type Ctrl-C
- An output summary appears listing the number of packets for each protocol and the number of alerts or logged packets.

Snort summary output

==== Initialization Complete ====

```
o^~  -> Snort! <*-  
o^~  )~  Version 2.6.1.5 (Build 59)  
o^~  ""  By Martin Roesch & The Snort Team: http://www.snort.org/team.html  
      (C) Copyright 1998-2007 Sourcefire Inc., et al.
```

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 1.6 <Build 11>

Not Using PCAP_FRAMES

*** Caught Int-Signal

Frag3 statistics:

```
Total Fragments: 0  
Frag Reassembled: 0  
Discards: 0  
Memory Faults: 0  
Timeouts: 0  
Overlaps: 0  
Anomalies: 0  
Alerts: 0  
FragTrackers Added: 0  
FragTrackers Dumped: 0  
FragTrackers Auto Freed: 0  
Frag Nodes Inserted: 0  
Frag Nodes Deleted: 0
```

```
=====  
Snort received 364 packets  
Analyzed: 181(49.725%)  
Dropped: 0(0.000%)  
Outstanding: 183( 50.275%)  
=====
```

```
Breakdown by protocol:  
TCP: 128 (70.718%)  
UDP: 42 (23.204%)  
ICMP: 0 (0.000%)  
ARP: 0 (0.000%)  
EAPOL: 0 (0.000%)  
IPv6: 0 (0.000%)  
ETHLOOP: 0 (0.000%)  
IPX: 0 (0.000%)  
FRAG: 0 (0.000%)  
OTHER: 11 (6.077%)  
DISCARD: 0 (0.000%)  
=====
```

```
Action Stats:  
ALERTS: 0  
LOGGED: 21  
PASSED: 0  
=====
```

Snort exiting



Snort's Pattern Matching

- Wu-manber algorithm (< v2.6)
- Aho-corsacik pattern matcher
 - Faster pattern matching
 - Requires more RAM



Snort Mailing Lists

- Snort rules

- <http://list.sourceforge.net/lists/listinfo/snort-sigs>

- Snort usage and application

- <http://lists.sourceforge.net/lists/listinfo/snort-users>



Interesting Paper

- Tim Newsham and Tom Ptacek
- "Insertion, Evasion, and Denial of Service: Evading Network Intrusion Detection"
- http://insecure.org/stf/secnet_ids/secnet_ids.html



Katie Smith

katie.smith@vandals.uidaho.edu