Intrusion Detection & Snort

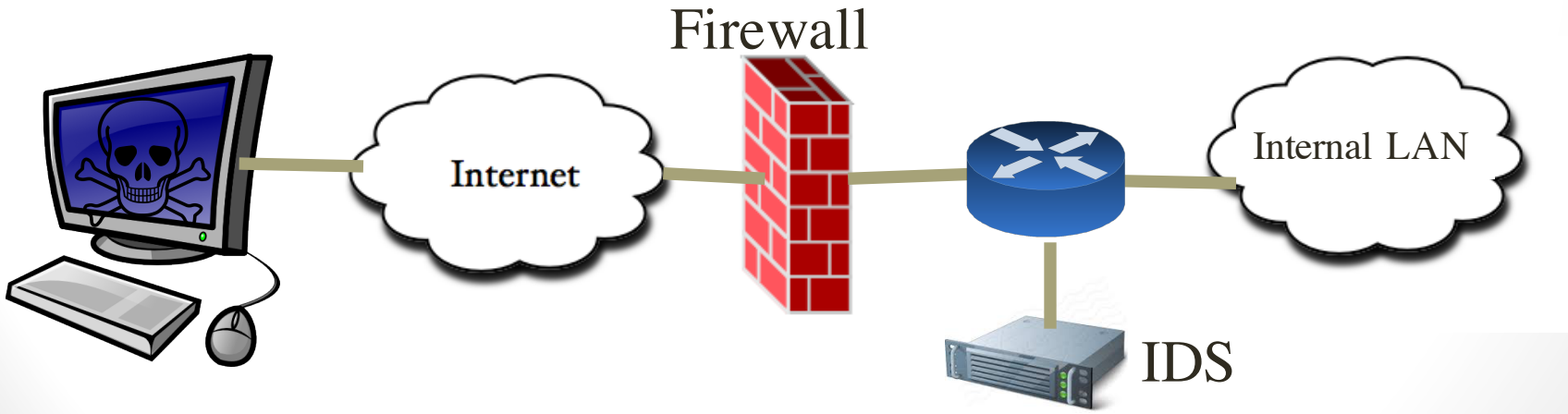
Dan Fleck, PhD

dfleck@gmu.edu

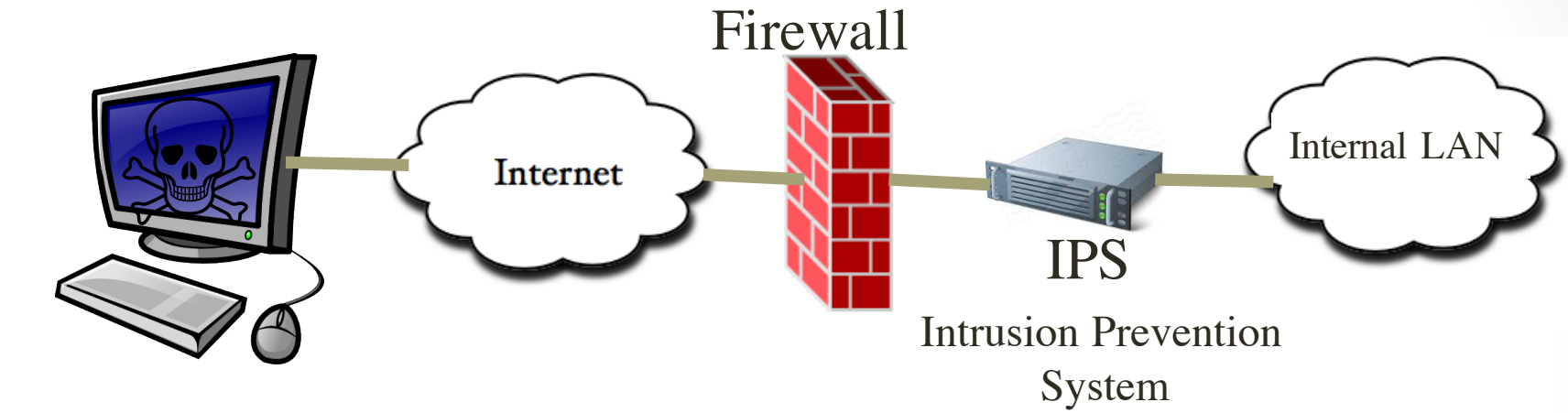
Intrusion Detection

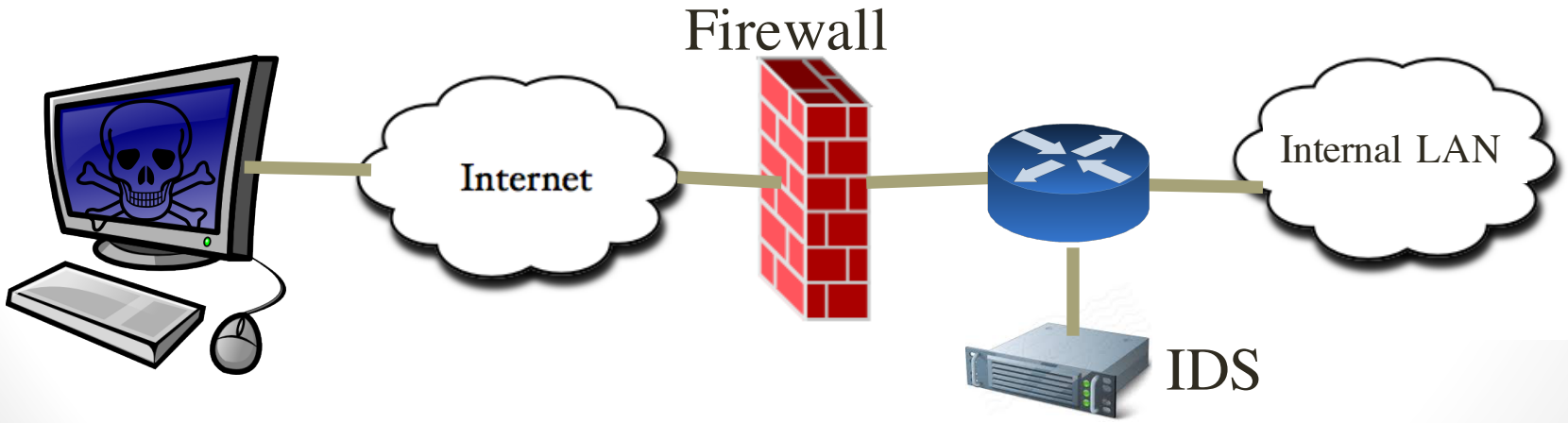
- An *intrusion detection system* (IDS) analyzes traffic patterns and reacts to anomalous patterns by sending out alerts.

- Note that an IDS is inherently reactive; the attack *has already begun* when the IDS alerts.



Intrusion Detection: IDS vs IPS





What changes if I want to see all attempted attacks?

Firewall vs IDS vs IPS

- **Firewall** - A device or application that analyzes packet headers and enforces policy based on protocol type, source address, destination address, source port, and/or destination port. Packets that do not match policy are rejected.

- **Intrusion Detection System** - A device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected a log message is generated detailing the event.

- **Intrusion Prevention System** - A device or application that analyzes whole packets, both header and payload, looking for known events. When a known event is detected the packet is rejected.

Some devices are now combining all of these functions into a single security device (Smart Firewall, Next Gen Firewall, etc…). Snort can be run in IDS or IPS modes.

Source: http://security.stackexchange.com/questions/44931/difference-between-ids-and-ips-and-firewall

What do IDS detect?

- **Anomaly detection**: Activity that deviates from the normal behavior
- **Misuse detection**: Execution of code that results in break-ins
- **Specifcation based detection**: Activity involving privileged software that is inconsistent with respect to a policy/specification

- D. Denning

Types of IDS

- Host Based IDS
  - Installed locally on machines
  - Monitoring local user activity
  - Monitoring execution of system programs
  - Monitoring local system logs
- Network IDS (NIDS)
  - Sensors are installed at strategic locations on the network
  - Monitor changes in traffic pattern/ connection requests
  - Monitor Users' network activity – Deep Packet inspection
- In this lab we're discussing NIDS

Types of NIDS

- Signature Based IDS
  - Compares incoming packets with known signatures
  - E.g. Snort, Bro, Suricata, etc.
- Anomaly Detection Systems
  - Learns the normal behavior of the system
  - Generates alerts on packets that are different from the normal behavior

Signature based NIDS

Current Standard is Signature Based Systems

Problems:

- “Zero-day” attacks
- Polymorphic attacks
- Botnets – Inexpensive re-usable IP addresses for attackers

Anomaly Detection NIDS

Anomaly Detection (AD) Systems are capable of identifying “Zero Day” Attacks

Problems:

- High False Positive Rates
- Labeled training data

Our Focus:

- Web applications are popular targets

transAD & STAND (GMU Research)

- transAD
  - TPR 90.17%
  - FPR 0.17%
- STAND
  - TPR 88.75%
  - FPR 0.51%

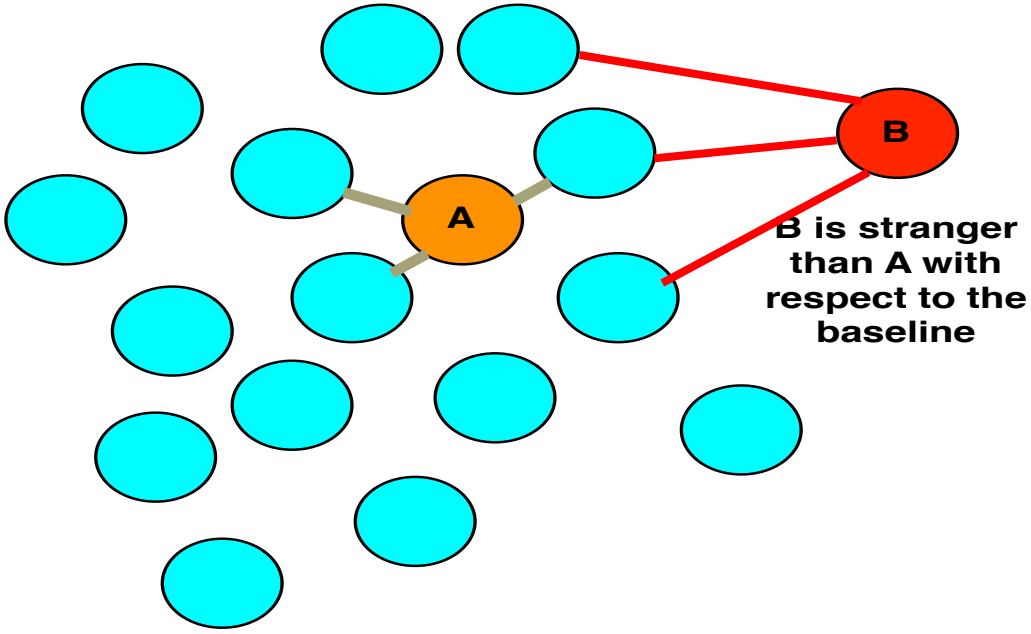
- What do you think a signature-based detector would look like (roughly) FPR? TPR?

Attacks Detected by transAD

Type of Attack	HTTP GET Request
Buffer Overflow	/?slide=kashdan?slide=pawloski?slide=ascoli?slide=shukla?slide=kabbani?slide=ascoli?slide=proteomics?slide=shukla?slide=shukla
Remote File Inclusion	//forum/adminLogin.php?config[forum installed]= http://www.steelcitygray.com/auction/uploaded/golput/ID-RFI.txt??
Directory Traversal	/resources/index.php?con=/../../../../../../../../etc/passwd
Code Injection	//resources-template.php?id=38-999.9+union+select+0
Script Attacks	/.well-known/autoconfig/mail/config-v1.1.xml?emailaddress=********%40*********.***.***

Transduction based Anomaly Detection

- Compares how test packet fits with respect to the baseline
- A “Strangeness” function is used for comparing the test packet
- The sum of K-Nearest Neighbors distances is used as a measure of Strangeness



Intrusion Detection Errors

There are two types of errors when considering any intrusion detection system.

False negatives: a genuine attack is not detected.
False positives: harmless behavior is misclassified as an attack.

Which do think is a bigger problem?

An intrusion detection system is:
accurate: if it detects all genuine attacks;
precise: if it never reports legitimate behavior as an attack.

It is easy to make an IDS that is either accurate or precise! *Why?*
It's hard to do both simultaneously.

Intrusion Detection Errors

An undetected attack might lead to severe problems. But frequent false alarms can lead to the system being disabled or ignored. A perfect IDS would be *both accurate and precise*.

- Statistically, attacks are fairly rare events.
- Most intrusion detection systems suffer from the *base-rate fallacy*.
- Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90% and the false positive rate is 10%. *If you have an alarm what is the chance it's a false alarm?*

Base-Rate Fallacy

Suppose that only 1% of traffic are actually attacks and the detection accuracy of your IDS is 90% and the false positive rate is 10%. *What does that mean?*

- the IDS classifies an attack as an attack with probability 90% (true positive)
- the IDS classifies a valid connection as attack with probability 10% (false positive)

What is the probability that a connection flagged as an attack is not really an attack, i.e., a false positive?

There is approximately 92% chance that a raised alarm is false.

Equations for Base Rate Fallacy

1000 events: 990 benign, 10 attacks.

10% False alarm rate means: 99 false alarms

90% True positive rate means: 9 true alarms

P(attack | alarm) = 9/(9+99) = 0.08%

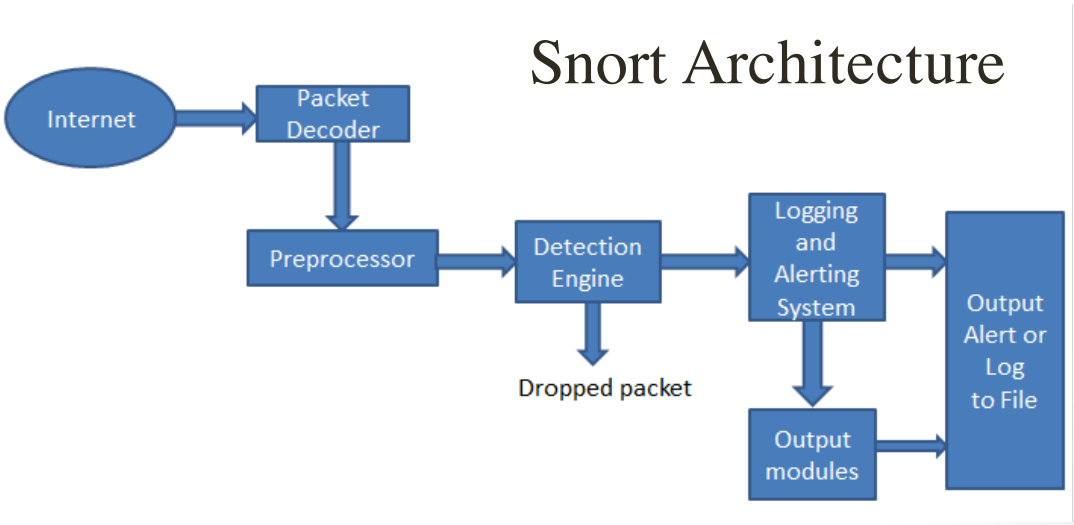
Meaning, 92% of alarms are false alarms due to the base rate of benign traffic. This is to give you intuition about base rate, this can be done more formally using Bayes rule.

Lessons

- False negatives and false positives are both bad for an IDS.
- An IDS must be very accurate or suffer from the base rate fallacy.
- An IDS with too many errors becomes useless.

Snort: Our lab

- Signature-based detection system
- 1 CPU w/ 1000 signatures can process 500MBps (not great!)
  - Getting faster in newer releases
- Can be run inline (IPS) or as a sniffer (IDS)
- First released in 1997 but still updated/maintained today
- Competitors: Suricata, Bro
- Detailed performance comparison: https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772



Snort: Rules

- http://manual.snort.org/node1.html
- http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-7-SECT-3.html

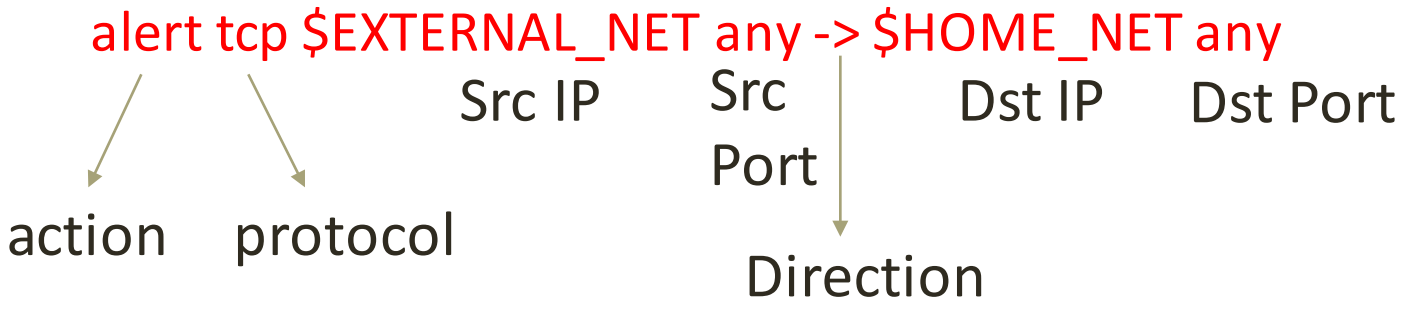
```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"SCAN SYN FIN";flags:SF; reference: arachnids,198; \
classtype:attempted-recon; sid:624; rev:1;)
```

rule header (rule options)

Snort: Rule Header

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"SCAN SYN FIN";flags:SF; reference: arachnids,198; \
classtype:attempted-recon; sid:624; rev:1;)
```

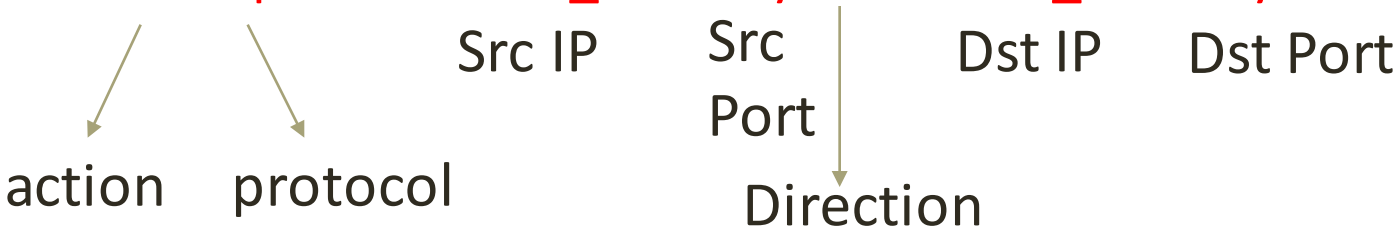
Defines “who” the rule applies to (coarsly).



Snort: Rule Header Actions

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"SCAN SYN FIN";flags:SF; reference: arachnids,198; \
classtype:attempted-recon; sid:624; rev:1;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
```

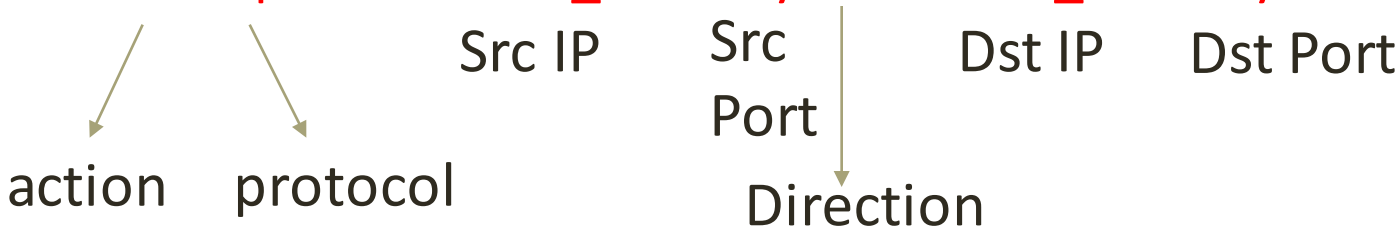


1. **alert**: Alerts and logs the packet when triggered.
2. **log**: Only logs the packet when triggered.
3. **pass** : Ignores or drops the packet or traffic matching.
4. **activate** : Alerts then activates a dynamic rule or rules.
5. **dynamic** : Ignores, until started by the activate rule, at which time, acts as a log rule.
6. **drop** : block and log the packet
7. **reject** : block the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
8. **sdrop** : block the packet but do not log it.

Snort: Rule Header Protocol

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"SCAN SYN FIN";flags:SF; reference: arachnids,198; \
classtype:attempted-recon; sid:624; rev:1;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
```



Protocols: TCP, UDP, ICMP, and IP
Future may include: ARP, IGRP, GRE, OSPF, RIP, IPX, etc.

Snort: Rule Header IP

Src Port

Src IP Dst IP Dst Port

alert tcp $EXTERNAL_NET any -> $HOME_NET any

alert tcp 192.168.1.0/24 any -> 192.168.1.0/24 1:1024

alert tcp ![192.168.1.0/24,10.1.1.0/24] any -> 192.168.1.44

$EXTERNAL_NET is a config value set in snort.conf.

IP is specified also as dotted notation with CIDR masks. “any” is also valid.

! is the negation operator

Multiple IP specifications can be included using square brackets [ ] and comma-separating.
Do not add spaces!

Snort: Rule Header Port

	Src IP	Src Port		Dst IP	Dst Port
alert tcp	$EXTERNAL_NET	any	->	$HOME_NET	any
alert tcp	192.168.1.0/24	any	->	192.168.1.0/24	1:1024
alert tcp	![192.168.1.0/24,10.1.1.0/24]	any	->	192.168.1.44	

Port can be specified as:

any	-- any port
1:1024	-- ports 1 to 1024 inclusive
55:	-- ports 55 and higher
:55	-- ports 0 to 55 (inclusive)

negation still works:

!6000:6001	- matches any port except 6000 and 6001

Snort: Rule Header Direction

	Src IP	Src Port		Dst IP	Dst Port
alert tcp	$EXTERNAL_NET	any	->	$HOME_NET	any
alert tcp	192.168.1.0/24	any	->	192.168.1.0/24	1:1024
alert tcp	![192.168.1.0/24,10.1.1.0/24]	any	->	192.168.1.44	

Direction can be specified as:

->	From right IP/Port (source) to left IP/Port (destination)
<>	Any direction

Note: <- does not exist… so the snort rules always read consistently.

Snort: Rule Options

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"SCAN SYN FIN";flags:SF; reference: arachnids,198;  \
classtype:attempted-recon; sid:624; rev:1;)
```

name:value;

msg: <sample message>	Logs message into /var/snort/log
flags: <AFPRSU210>	Matches specific TCP flags
content: <text>	Matches specified text in packet
content: |<hexadecimal>|	Matches specified hex chars
sid: <snort ID>	Unique number to identify rules easily. Your rules should use SIDs > 1,000,000
rev: <revision #>	Rule revision number
reference:<ref>	Where to get more info about the rule
gid:<generator ID>	Identifies which part of Snort generated the alert. See /etc/snort/gen-msg.map for values

Snort: More Rule Options…

Read the docs.. there are MANY more options:

http://manual.snort.org/node1.html

Snort rule examples

1. alert tcp any any -> any 21 (flow:to_server,established; \
   content:"root"; pcre:"/user\s+root/i";)

   What does it do?

Looks for root user login attempts on FTP server (port 21)

Snort: Try it out!

- Lets build two new rules to see how they work
- Rule 1: Alert if a URI is longer than 250 bytes.
- Rule 2: Alert on .edu websites that also say “university” in the page somewhere. (Because we love school!!)

  Wouldn’t this be more fun in IPS mode? ☺

Snort rule examples

This is a real rule from malware-tools.rules

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"MALWARE-TOOLS HOIC http denial of service
attack"; flow:to_server,established; content:"User-
Agent|3A 20 20|Mozilla"; fast_pattern:only;
http_header; content:"Referer|3A 20 20|http";
http_header; content:!"Connection: keep-alive"; nocase;
detection_filter:track by_src, count 17, seconds 10;
metadata:policy balanced-ips drop, policy security-ips
drop, service http;
reference:url,blog.spiderlabs.com/2012/01/hoic-ddos-
analysis-and-detection.html; classtype:denial-of-
service; sid:21513; rev:6;)
```

Snort rule examples

This is a real rule from blacklist.rules

```
alert udp $HOME_NET any -> any 53 (msg:"BLACKLIST DNS
request for known malware domain guest-access.net -
Gauss "; flow:to_server; byte_test:1,!&,0xF8,2;
content:"|0C|guest-access|03|net|00|";
fast_pattern:only; metadata:impact_flag red, policy
balanced-ips drop, policy security-ips drop, service
dns; reference:url,gauss.crysys.hu/;
reference:url,www.securelist.com/en/blog/208193767/Gaus
s_Nation_state_cyber_surveillance_meets_banking_Trojan;
classtype:trojan-activity; sid:23799; rev:2;)
```

Snort rule examples

This is a real rule from os-windows.rules

```
alert tcp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET
any (msg:"OS-WINDOWS Microsoft Windows Object Packager
ClickOnce object remote code execution attempt";
flow:to_client,established;
flowbits:isset,file.ppsx&file.zip; file_data;
content:"uuid:48fd9e68-0958-11dc-9770-9797abb443b9";
fast_pattern:only; content:"2007-05-23T15:06:10-03:00";
metadata:policy balanced-ips drop, policy security-ips
drop, service ftp-data, service http, service imap,
service pop3; reference:cve,2012-0013;
reference:url,technet.microsoft.com/en-
us/security/bulletin/ms12-005; classtype:attempted-
user; sid:26068; rev:3;)
```

For the lab…

- Put your rules in: /etc/snort/rules/local.rules
- The rules included in the default download are old and terrible. To really play with Snort you need a current ruleset. One place to get them is snort.org
- The nocase option is a content-modifier to ignore case. Put it right after content it should modify:

```
alert tcp $EXTERNAL_NET any -> $TELNET_SERVERS 23 ( sid: 210; rev: 3;
msg: "BACKDOOR attempt"; flow: to_server,established; content:
"backdoor"; nocase; classtype: attempted-admin;)
```

- Remember that payload rules don't work on encrypted traffic! (SSL, etc..)
- Use "–A console" to debug alerts on the console
- Use "–k none" to disable tcp checksums

References

- http://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm#Basics
- http://www.scmagazine.com/intrusion-detection-systems/products/91/0/
- http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-7-SECT-3.html
- http://seclists.org/snort/2012/q3/894