Attack Detection and Response with Linux Firewalls

Michael Rash
Security Architect
Enterasys Networks, Inc.

http://www.cipherdyne.org/

03/25/2007
ShmooCon, 2007

Agenda

- Why use iptables to detect attacks?
- Types of attacks that can be detected
  - Snort rules language emulation via iptables logs and string match extension
- psad and fwsnort (http://www.cipherdyne.org/)
- Honeynet iptables log data visualizations
  - psad + AfterGlow
- Live Demo

Supplementing IDS Infrastructure?

- Snort and commercial IDS infrastructure is mature (subject to usual concerns around false positives), but why stop there?
- IDS's can themselves be targeted, both from the detection and code execution standpoints
  - Modified Stick/Snot to send faked attacks over Tor
  - Snort DCE/RPC Preprocessor vulnerability
- Defense-in-depth is important
- Host fragment reassembly issues less of a concern for iptables string matching (more on this later)

Major IDS Functionality We Need

- Signature based packet inspection
  - packet headers
  - application layer data
- Network and transport layer reassembly*
  - Ideally, reassembly algorithms would be identical to target host
- Reporting
- Ability to respond to attacks (optional)

Snort Signature Language

- Widely accepted and deployed IDS language
- Updated signatures readily available; http://www.bleedingsnort.com (BSD-licensed)
- Vulnerability announcement services (iDefense, Endeavor Security) provide signatures in Snort format
- Defines a good direction for iptables emulation

IDS and iptables

- Can specify granular packet header tests, and logging format contains nearly all interesting packet header fields
- Can match against connection states
  - Useful for mitigating Stick/Snot style attacks
- String matching in the kernel started in the 2.4 days (patch applied via Netfilter patch-o-matic); made available again in 2.6.14

IDS and iptables (cont'd)

- Kernel textsearch (linux/lib/ts_*) infrastructure
  - Boyer-Moore and Knuth-Morris-Pratt algorithms
- String matching enabled by default in recent Linux kernels
- You get network layer defragmentation for free when connection tracking is used – you don't have to rely on proper configuration of frag3; it ***is*** the defragmentation algorithm of the host
- String matching within the *filter* table happens after network defrag

Active Response / Intrusion Prevention

- Plenty of reasons *NOT* to respond (false positives, possibility of attacker abuse, possibility of fingerprinting the response mechanism)
- However:
  - Can envision scenarios where controlling the shape of application layer data that can talk to local sockets is a good thing – iptables can enforce the DROP target (this is ***prevention*** instead of just some weak ***response*** mechanism)
  - Some automated attacks do not bother with obfuscation/encryption – target rich environment
  - Sometimes it is not easy to patch a production server whose uptime must remain high (assuming a patch even exists)

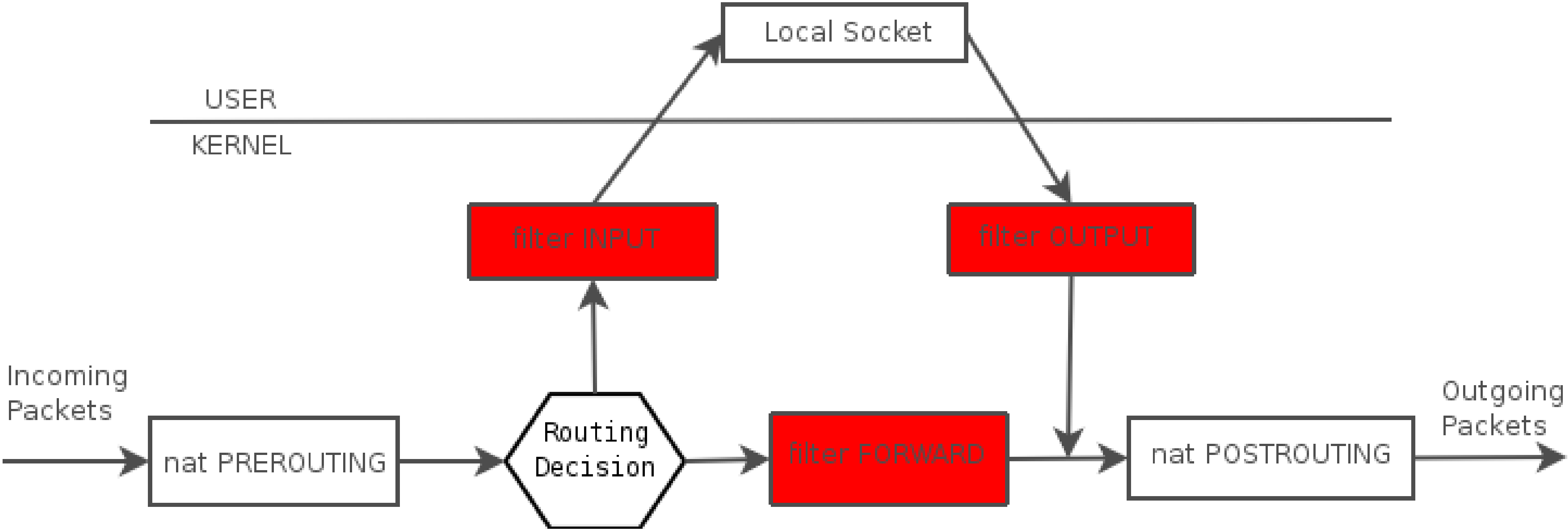
fwsnort

- Translates Snort signatures into “equivalent” iptables rules using string match extension and Netfilter connection tracking subsystem
- All translated Snort signatures are placed within user-defined chains, to which packets are jumped from built-in chains (INPUT, OUTPUT, and FORWARD)
- Maintains strict separation from existing iptables policy
- Approximately 60% of all Snort-2.3.3 rules (remember this is an IDS supplement) can be translated

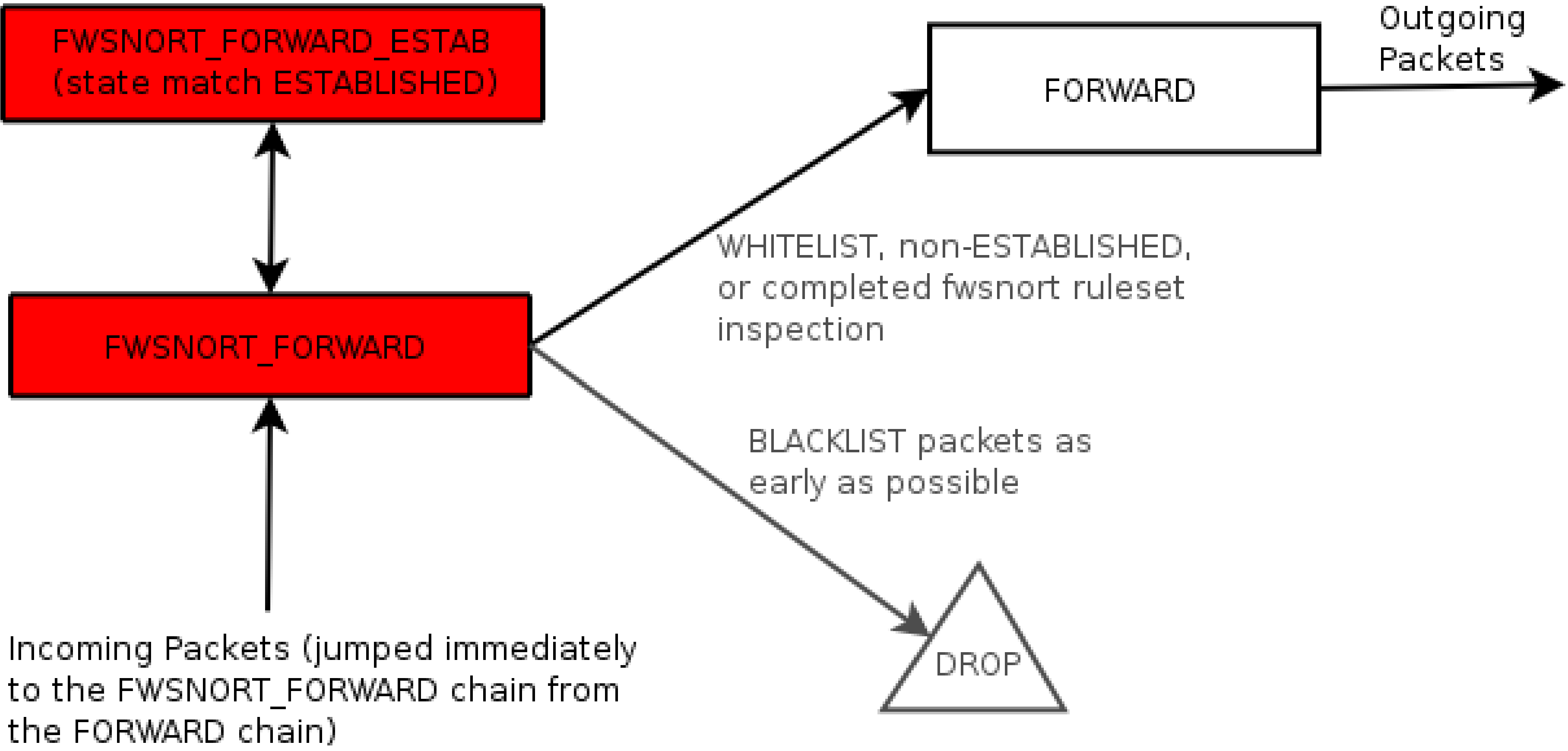
fwsnort (cont'd)

- Emulation of Snort config variables such as $HOME_NET and $EXTERNAL_NET
- Reporting via LOG target (integrates with psad)
- Whitelists via the RETURN target
- Blacklists via the DROP or REJECT targets
- Snort signature info stored with the iptables comment match in kernel-space (0.9.0 release)
- iptables is inline by definition; easy to configure fwsnort to use the DROP or REJECT targets

iptables Packet Flow



fwsnort Packet Flow



FWSNORT_FORWARD_ESTAB (state match ESTABLISHED)

FORWARD

Outgoing Packets

FWSNORT_FORWARD

WHITELIST, non-ESTABLISHED, or completed fwsnort ruleset inspection

BLACKLIST packets as early as possible

DROP

Incoming Packets (jumped immediately to the FWSNORT_FORWARD chain from the FORWARD chain)

Example Snort Rule: nmap Execution via Web Server

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS

(msg:"WEB-ATTACKS nmap command attempt"; **flow**:to_server,established; **content**:"nmap%20"; nocase; classtype:web-application-attack; sid:1361; rev:5;)

fwsnort Translation

```
$IPTABLES -A FWSNORT_FORWARD_ESTAB -d
192.168.10.0/24 -p tcp --dport 80 -m string
--string "nmap%20" --algo bm -m comment --
comment "msg: WEB-ATTACKS nmap command
attempt; classtype: web-application-attack;
rev: 5; FWS:0.9.0;" -j LOG --log-tcp-
options --log-prefix "[1] SID1361 ESTAB "
```

“BLEEDING-EDGE VIRUS” Signature (Multiple Content Fields)

```
alert tcp $EXTERNAL_NET $HTTP_PORTS ->
$HOME_NET any (msg: "BLEEDING-EDGE VIRUS
Trojan-Spy.Win32.Bancos Download"; flow:
established,from_server; content:"[AspackDie!]";
content:"|0f 6d 07 9e 6c 62 6c 68 00 d2 2f 63 6d 64 9d 11
af af 45 c7 72 ac 5f 3138 d0|"; classtype: trojan-activity;
reference:url,securityresponse.symantec.com/avcenter/ve
nc/data/pwsteal.bancos.b.html; sid: 2001726; rev:6; )
```

(translated)

```
$IPTABLES -A FWSNORT_FORWARD_ESTAB -d
192.168.10.0/24 -p tcp --sport 80 -m string --string
"[AspackDie!]" --algo bm -m string --hex-string "|0f 6d
07 9e 6c 62 6c 68 00 d2 2f 63 6d 64 9d 11 af af 45 c7 72
ac 5f 3138 d0|" --algo bm -m comment --comment "msg:
BLEEDING-EDGE VIRUS Trojan-Spy.Win32.Bancos
Download; classtype: trojan-activity; reference:
url,securityresponse.symantec.com/avcenter/venc/data/pw
steal.bancos.b.html; rev: 6; FWS:0.9.0;" -j LOG --log-ip-
options --log-tcp-options --log-prefix "[640] SID2001726
ESTAB "
```

“Matrix 2.0 Server Access” Signature

```
alert udp $EXTERNAL_NET 3345 -> $HOME_NET 3344
(msg:"BACKDOOR Matrix 2.0 Server access"; content:"|
0b ab ab ab 00 00 00 |logged in"; reference:arachnids,83;
classtype:misc-activity; sid:162; rev:4;)
```

(translated)

```
$IPTABLES -A FWSNORT_FORWARD -d
192.168.10.0/24 -p udp --sport 3345 --dport
3344 -m string --hex-string "|0b ab ab ab
00 00 00 |logged in" --algo bm -m comment
--comment "msg: BACKDOOR Matrix 2.0 Server
access; classtype: misc-activity;
reference: arachnids,83; rev: 4;
FWS:0.9.0;" -j LOG --log-ip-options --log-
prefix "[46] SID162 "
```

“IPSec PGPNet Connection Attempt” Signature

```
alert udp $EXTERNAL_NET any -> $HOME_NET 500
(msg:"POLICY IPSec PGPNet connection attempt";
content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
10 02 00 00 00 00 00 00 00 00 88 0D 00 00 5C 00 00 00
01 00 00 00 01 00 00 00|P|01 01 00 02 03 00 00 24 01 01
00 00 80 01 00 06 80 02 00 02 80 03 00 03 80 04 00 05
80 0B 00 01 00 0C 00 04 00 01|Q|80 00 00 00 24 02 01 00
00 80 01 00 05 80 02 00 01 80 03 00 03 80 04 00 02 80
0B 00 01 00 0C 00 04 00 01|Q|80 00 00 00 10|";
classtype:protocol-command-decode; sid:1771; rev:6;)
```

(translated)

```
$IPTABLES -A FWSNORT_FORWARD -d 192.168.10.0/24
-p udp --dport 500 -m string --hex-string "|00
00 00 00 00 00 00 00 00 00 00 00 00 00 01 10 02
00 00 00 00 00 00 00 00 88 0D 00 00 5C 00 00 00
01 00 00 00 01 00 00 00|P|01 01 00 02 03 00 00
24 01 01 00 00 80 01 00 06 80 02 00 02 80 03 00
03 80 04 00 05 80 0B 00 01 00 0C 00 04 00 01|Q|
80 00 00 00 24 02 01 00 00 80 01 00 05 80 02 00
01 80 03 00 03 80 04 00 02 80 0B 00 01 00 0C 00
04 00 01|Q|80 00 00 00 10|" --algo bm -m
comment --comment "msg: POLICY IPSec PGPNet
connection attempt; classtype: protocol-
command-decode; rev: 6; FWS:0.9.0;" -j LOG --
log-ip-options --log-prefix "[1005] SID1771 "
```

Supported Snort Rule Options

- All Snort rule header options
- content
- flow (conntrack)
- flags
- offset
- depth
- dsize (length match)
- itype
- icode
- ttl (ttl match)
- tos (tos match)
- ipopts
- ip_proto
- resp

Unsupported Snort Rule Options: Lost in Translation

- pcre
- flowbits
- byte_test <-- u32 module (need a 2.6 port)
- byte_jump <-- u32 module (need a 2.6 port)
- asn1
- window <-- included in iptables logs
- isdataat
- id <-- included in iptables logs

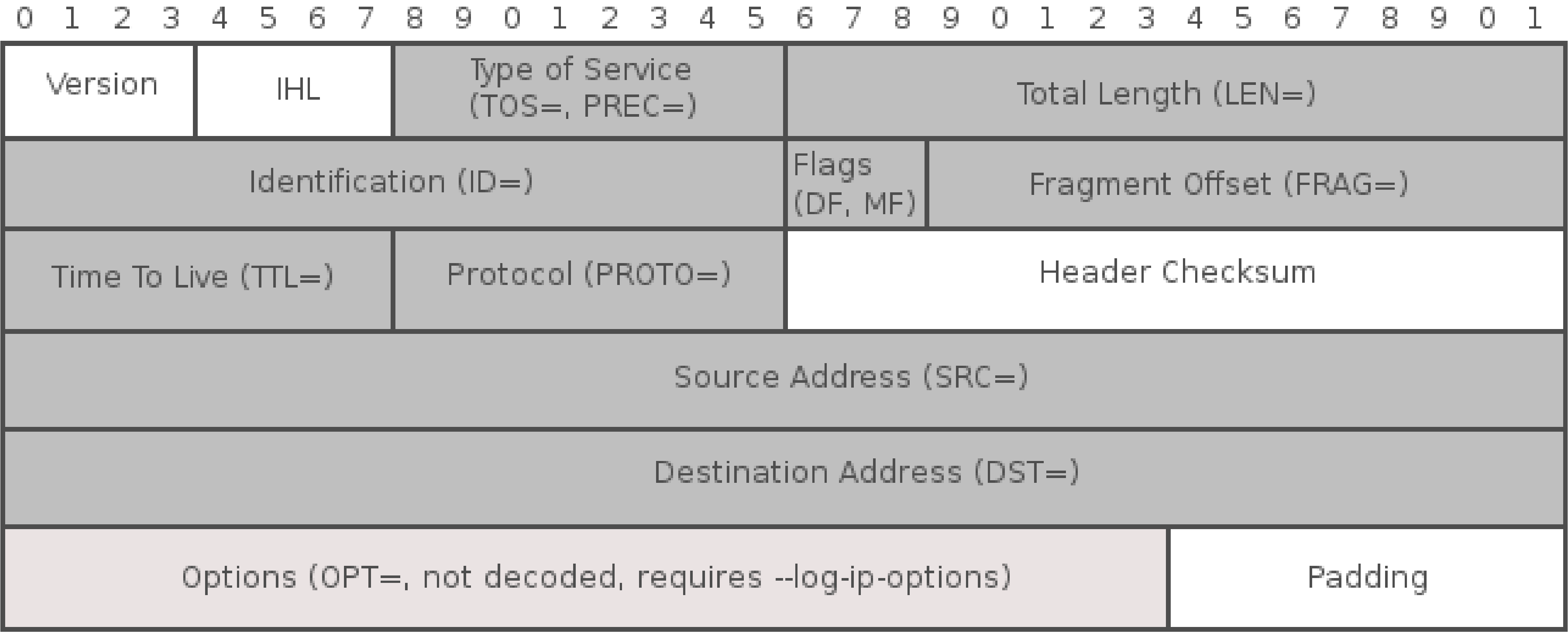
Unsupported Snort Rule Options (cont'd)

- icmp_id <-- included in iptables logs
- icmp_seq <-- included in iptables logs
- seq <-- included with --log-tcp-sequence
- ack <-- included with --log-tcp-sequence
- sameip <-- included in iptables logs
- There are a few others - those that are logged can be analyzed by psad

iptables TCP Log Message

```
Mar 11 20:21:22 minastirith kernel: [199]
SID1361 ESTAB IN=eth1 OUT=
MAC=00:13:d3:38:b6:e4:00:13:46:c2:60:44:08:
00 SRC=192.168.10.3 DST=192.168.10.1 LEN=60
TOS=0x00 PREC=0x00 TTL=63 ID=11112 DF
PROTO=TCP SPT=28778 DPT=80 WINDOW=5840
RES=0x00 ACK PSH URGP=0 OPT
(0101080A02A041D20CC386B1)
```

iptables IP Header Coverage



0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Version

IHL

Type of Service (TOS=, PREC=)

Total Length (LEN=)

Identification (ID=)

Flags (DF, MF)

Fragment Offset (FRAG=)

Time To Live (TTL=)

Protocol (PROTO=)

Header Checksum

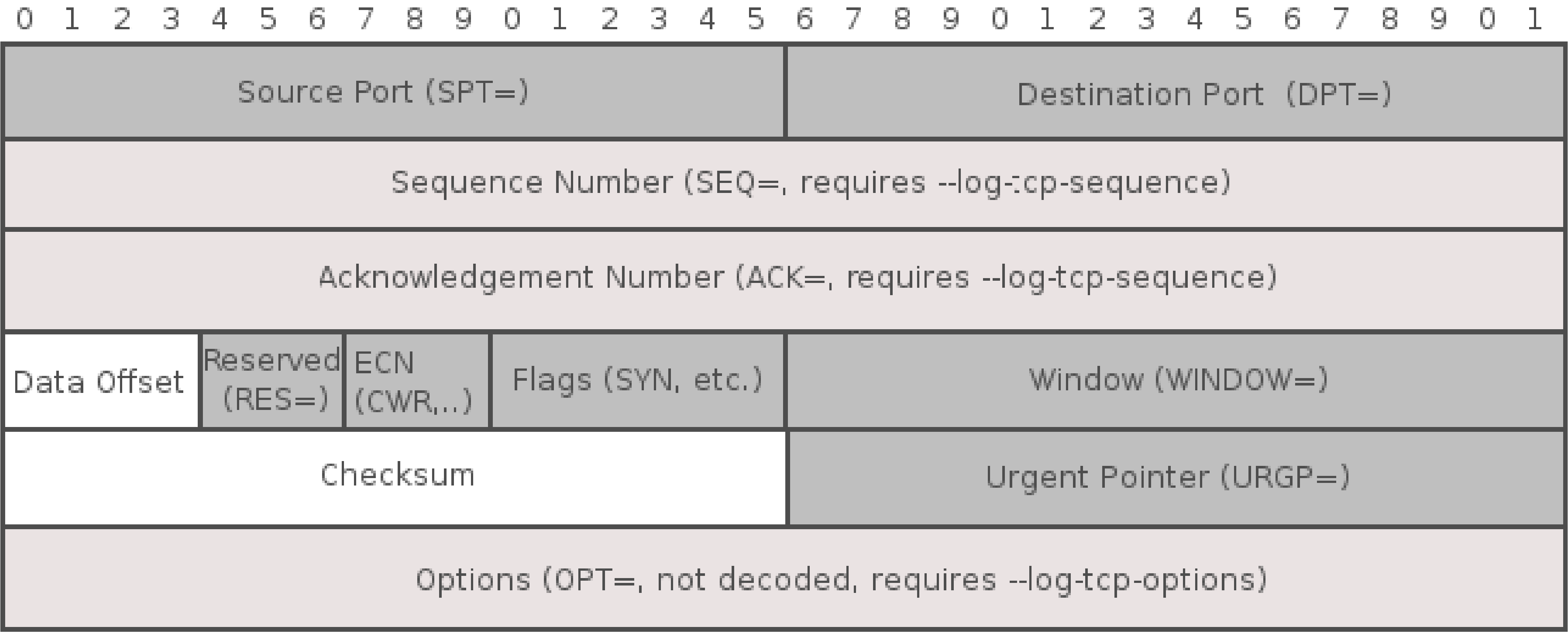
Source Address (SRC=)

Destination Address (DST=)

Options (OPT=, not decoded, requires --log-ip-options)

Padding

iptables TCP Header Coverage

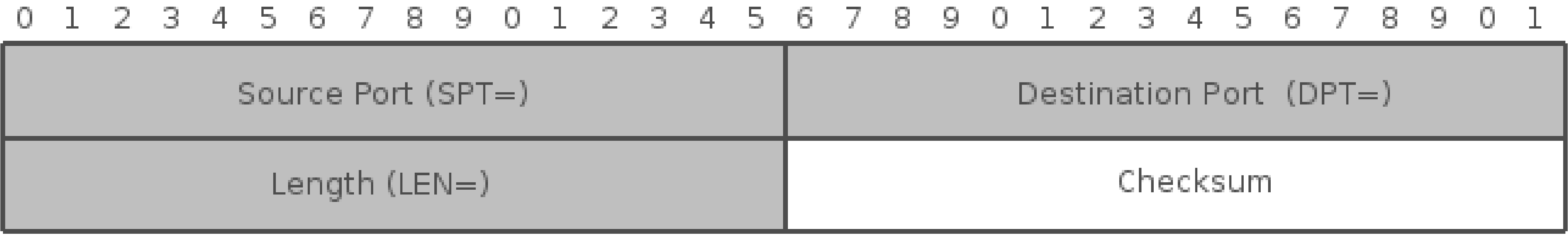


<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="16">Source Port (SPT=)</td><td colspan="16">Destination Port (DPT=)</td></tr>
<tr><td colspan="32">Sequence Number (SEQ=, requires --log-tcp-sequence)</td></tr>
<tr><td colspan="32">Acknowledgement Number (ACK=, requires --log-tcp-sequence)</td></tr>
<tr><td colspan="4">Data Offset</td><td colspan="3">Reserved
(RES=)</td><td colspan="3">ECN
(CWR,..)</td><td colspan="6">Flags (SYN, etc.)</td><td colspan="16">Window (WINDOW=)</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent Pointer (URGP=)</td></tr>
<tr><td colspan="32">Options (OPT=, not decoded, requires --log-tcp-options)</td></tr>
</table>

iptables UDP Log Message

```
Mar 11 20:50:54 minastirith kernel: [153]
SID2001597 IN=eth0 OUT=
MAC=00:13:d3:38:b6:e4:00:13:46:c2:60:44:08:
00 SRC=192.168.10.3 DST=192.168.10.1 LEN=40
TOS=0x00 PREC=0x00 TTL=63 ID=29758 DF
PROTO=UDP SPT=32046 DPT=61 LEN=20
```

iptables UDP Header Coverage



0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Source Port (SPT=)

Destination Port (DPT=)

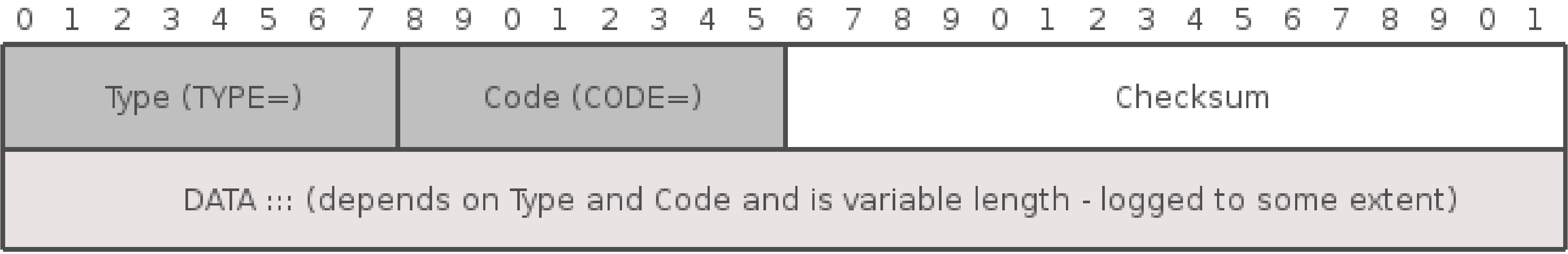
Length (LEN=)

Checksum

iptables ICMP Log Message

```
Mar 11 20:57:18 minastirith kernel: [98]
SID2003294 IN=eth0 OUT=
MAC=00:13:d3:38:b6:e4:00:13:46:c2:60:44:08:
00 SRC=192.168.10.3 DST=192.168.10.1
LEN=128 TOS=0x00 PREC=0x00 TTL=63 ID=53466
PROTO=ICMP TYPE=8 CODE=0 ID=27459 SEQ=0
```

iptables ICMP Header Coverage



0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Type (TYPE=)

Code (CODE=)

Checksum

DATA ::: (depends on Type and Code and is variable length - logged to some extent)

psad

- iptables log analyzer
- Email and syslog reporting
- Fwsnort integration
- Dshield integration
- iptables LOG visualization with AfterGlow
- Built-in passive OS fingerprinting derived from p0f (requires --log-tcp-options)
- IP options decoding (requires --log-ip-options)

psad (cont'd)

- Can detect Snort signatures that do not require application layer tests (source routing attempts, low ttl values, ICMP source quench, Nachi worm, etc.). This is all possible by virtue of iptables LOG format completeness.
- Detection of many port scan types generated by Nmap
- Timeout-based auto-blocking (optional, and can be restricted to application layer matches with fwsnort)
- Whitelists/Blacklists

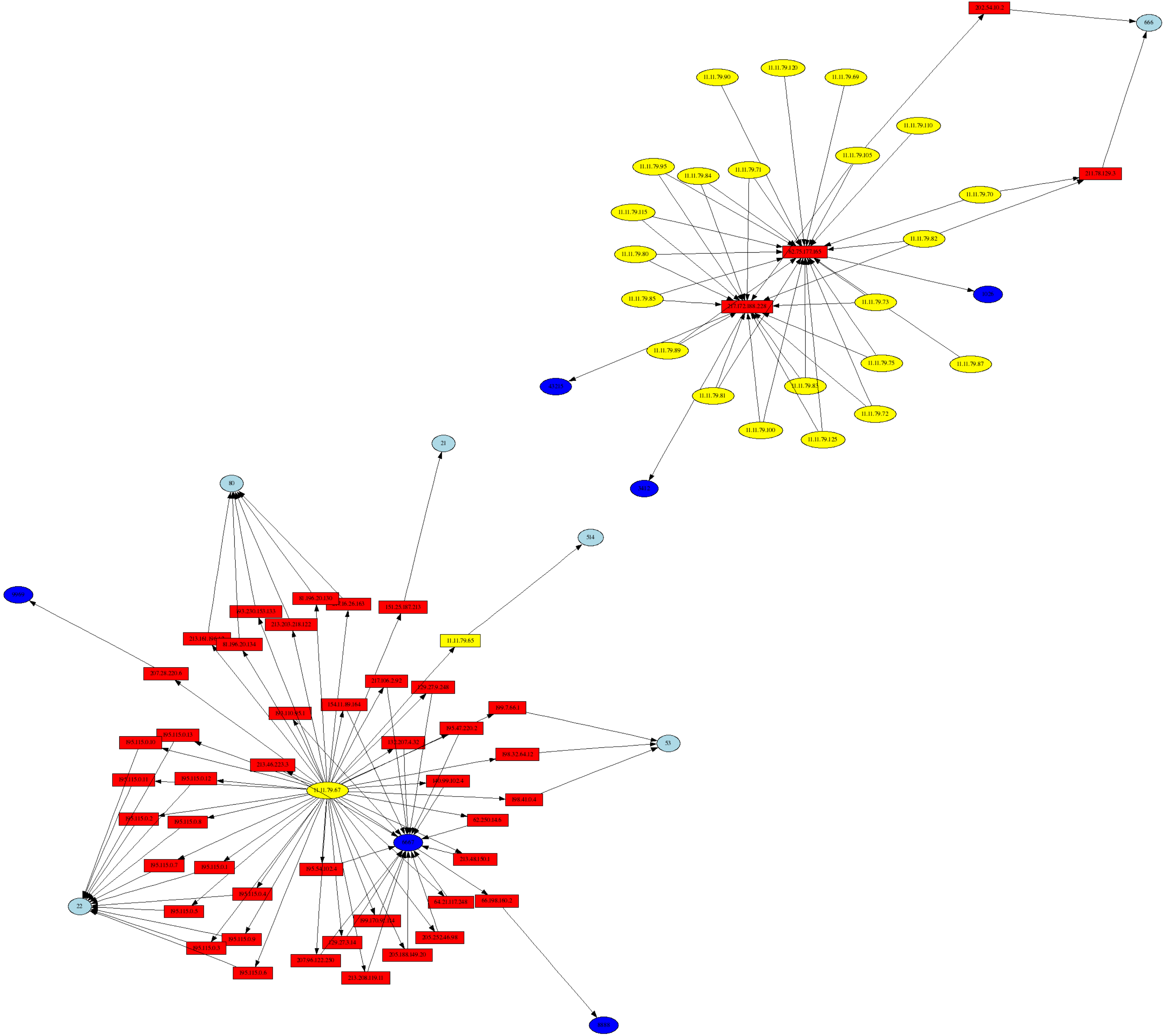
Honeynet Scan Challenge #34

- Challenge summary:
  - Challenge information and analysis can be found here: http://www.honeynet.org/scans/scan34/
  - Honeynet systems were compromised
  - Both Snort and iptables log data made available to the community (39MB of iptables data)
  - Honeynet IP addresses sanitized to 11.11.x.x
  - BTW, check out http://www.secviz.org

Honeynet Visualizations: Compromised Hosts

- Look for outbound connections from honeynet hosts:

```
# psad --CSV -m iptablessyslog --CSV-fields
"src:11.11.79.0/24 dst dp" | perl
afterglow.pl -c color.properties |neato
-Tgif -o outbound_connections.gif
```

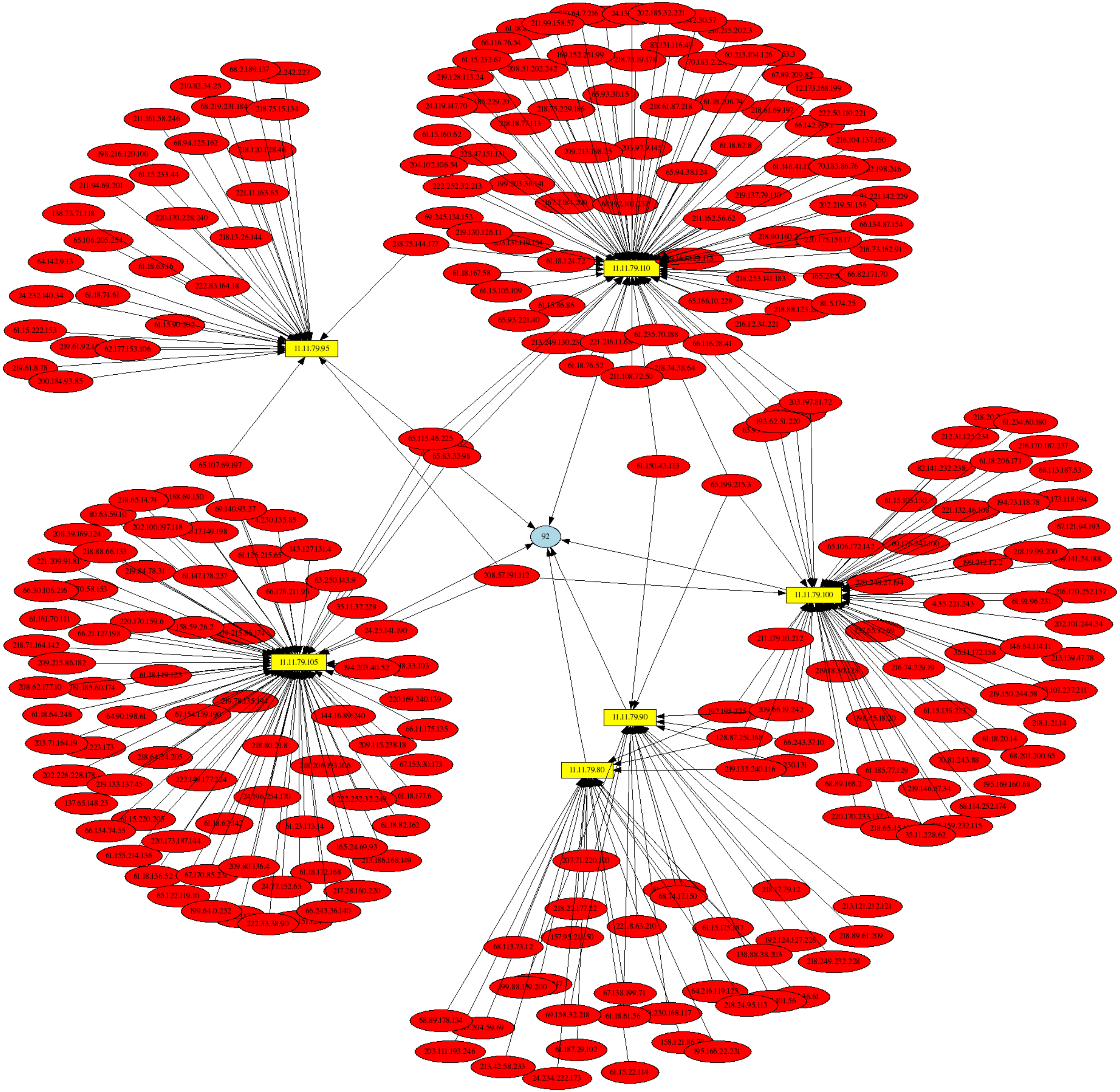


202.54.10.2
666
11.11.79.120
11.11.79.90
11.11.79.69
11.11.79.110
11.11.79.105
211.78.129.3
11.11.79.95
11.11.79.84
11.11.79.71
11.11.79.70
11.11.79.115
11.11.79.82
11.11.79.80
62.75.177.165
1026
11.11.79.85
217.172.188.228
11.11.79.73
11.11.79.89
11.11.79.75
11.11.79.87
43215
11.11.79.83
11.11.79.81
11.11.79.72
11.11.79.100
11.11.79.125
3412
21
80
514
9969
81.196.20.130
217.16.26.163
151.25.187.213
193.230.153.133
213.203.218.122
213.161.196.1
81.196.20.134
11.11.79.65
207.28.220.6
217.106.2.92
129.27.9.248
154.11.89.164
193.110.95.1
199.7.66.1
195.115.0.13
195.115.0.10
195.47.220.2
132.207.4.32
53
213.46.223.3
198.32.64.12
195.115.0.11
195.115.0.12
140.99.102.4
11.11.79.67
198.41.0.4
195.115.0.2
195.115.0.8
62.250.14.6
6667
213.48.150.1
195.115.0.7
195.115.0.1
195.54.102.4
195.115.0.4
64.21.117.248
66.198.160.2
22
195.115.0.5
199.170.91.114
195.115.0.9
129.27.3.14
205.252.46.98
195.115.0.3
207.96.122.250
205.188.149.20
195.115.0.6
213.208.119.11
8888

Nachi Worm Visualization

- Look for 92-byte ICMP echo requests

```
# psad --CSV -m iptablessyslog --CSV-fields
"src dst ip_len:92" --CSV-max 300 --CSV-regex
"PROTO=ICMP.*TYPE=8" | perl afterglow.pl -c
color.properties |neato -Tgif -o
nachi_worm.gif
```

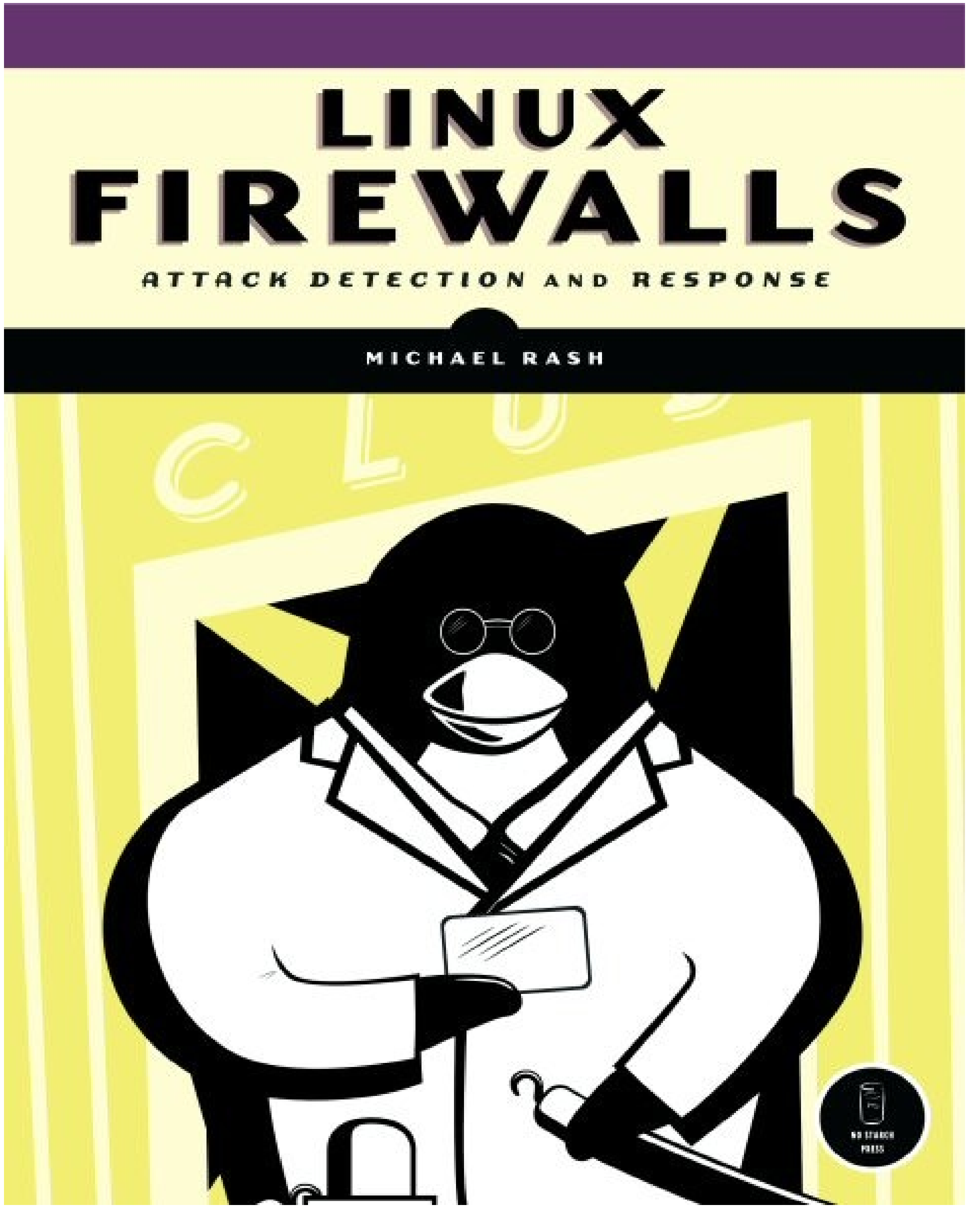

11.11.79.110
11.11.79.95
11.11.79.105
11.11.79.100
11.11.79.90
11.11.79.80
92
218.75.144.177
65.107.69.197
65.115.46.225
65.83.33.98
208.57.191.112
61.150.43.113
65.199.215.3
203.197.81.72
193.62.51.220
24.25.141.190

Honeynet Project Requests

- If providing iptables log data, please:
  - Use --log-ip-options
  - Use --log-tcp-sequence
  - Use --log-tcp-options
  - More attacks can be detected, and operating systems can be passively fingerprinted

Live Demo...

No Starch Press, June 2007



Questions?

http://www.cipherdyne.org/

mbr@cipherdyne.org