Advanced Netfilter; Content Replacement (ala Snort_inline) and Port Knocking Based on p0f

DEFCON 12

Michael Rash

http://www.cipherdyne.org
http://www.enterasys.com/

08/01/2004

Introduction

- Iptables logging format
- Passive OS fingerprinting with p0f
- What can iptables tell us?
- fwknop
- Iptables string match extension
- Snort_inline
- String replacement patch
- Netperf benchmarks

Iptables Logs

```
iptables -A INPUT -p tcp -i eth0 -j DROP --log-prefix “DROP “ --log-tcp-options
```

Iptables Logs; Decoded IP header fields

- Source and destination IP addresses
- IP datagram length
- Type of service
- TTL
- IP ID
- Fragment bits
- Protocol

```
Jul  8 03:06:12 orthanc kernel: DROP IN=eth0 OUT=
MAC=00:a0:cc:28:42:5a:00:00:00:22:2d:42:00:00
SRC:192.168.10.3 DST:192.168.10.1 LEN=60 TOS=0x10
PREC=0x00 TTL=64 ID=6854 DF PROTO=TCP
```

Iptables Logs; Decoded TCP header fields

- Source and destination ports
- TCP window size
- TCP flags

Jul 8 03:06:12 orthanc kernel: DROP IN=eth0 OUT= MAC=00:a0:cc:28:42:5a:00:00:00:22:2d:42:00:00 SRC:192.168.10.3 DST:192.168.10.1 LEN=60 TOS=0x10 PREC=0x00 TTL=64 ID=6854 DF PROTO=TCP SPT=32788 DPT=5500 WINDOW=5840 RES=0x00 SYN URGP=0

Iptables Logs; Encoded TCP header fields

- TCP options!!! p0f depends on this.

  Jul 8 03:06:12 orthanc kernel: DROP IN=eth0 OUT= MAC=00:a0:cc:28:42:5a:00:00:00:22:2d:42:00:00 SRC:192.168.10.3 DST:192.168.10.1 LEN=60 TOS=0x10 PREC=0x00 TTL=64 ID=6854 DF PROTO=TCP SPT=32788 DPT=5500 WINDOW=5840 RES=0x00 SYN URGP=0 OPT (020405B40402080A006F1D8E00000000001030300)

TCP options; What does p0f Care About?

- MSS (Maximum Segment Size)
- Window scale
- Selective acknowledgement (permitted bits)
- NOP
- Timestamp

  Encoding: type (8 bits) / total length (8 bits) / value (n - 16 bits)

  e.g. 020405b4 = MSS / 4 bytes / 1460

Decoded TCP options

- OPT (020405B40402080A00749E8600000000001030300)
  - MSS = 1460
  - Selective Acknowledgement permitted
  - Timestamp
  - NOP
  - Window Scale = 0

Packet Summary

- Length = 60
- Don't fragment bit
- TTL = 64
- Window size = 5840
- MSS = 1460
- Selective Acknowledgement permitted
- Timestamp
- NOP
- Window Scale = 0

What does p0f have to say?

S4:64:1:60:M*,S,T,N,W0

Linux:2.4::Linux 2.4/2.6

Other fingerprinting strategies

- IP ID
- Type of Service

"Passive OS Fingerprinting: Details and Techniques", Toby Miller

XProbe

Port Knocking

- Information hiding within sequences of connections to closed (or open) ports.
- Access control modification.
- Can be encrypted or shared.
- Multiple protocols.
- Relative timings.
- Third party IP access.

http://www.portknocking.org, MartinKrzywinski

fwknop

- Iptables log messages.
- Shared or encrypted knock sequences.
- Multi-protocol (tcp and udp)
- Relative and absolute port timings.
- p0f
- Exact or regex OS match.

Implementation

- “Client/Server”
- Knock sequences encrypted via Rijndael
- Syslog monitor “knopmd”
- named pipe
- sysklogd and syslog-ng
- OpenBSD pf.os
- Selectable iptables ruleset entry
- Access timeout and iptables connection tracking.

Live Demo...

Iptables string match extension

- Application layer inspection
- Boyer-Moore algorithm
- BM_MAX_HLEN = 1024, netfilter_ipv4/ipt_string.h

String match interface

Snort SID 940: “WEB-FRONTPAGE shtml.dll

```
iptables -I FORWARD 1 -p tcp --dport 80 --
tcp-flags ACK ACK -m string  --string
“/_vti_bin/shtml.dll” -j LOG --log-prefix
“SID940 ”
```

String match interface (2)

Snort SID 261: “DNS EXPLOIT named overflow attempt”

```
iptables -I FORWARD 1 -p tcp --dport 53 --tcp-flags ACK ACK -m string --hex-string “|CD80 E8D7 FFFF FF|/bin/sh” -j LOG --log-prefix “SID261 ”
```

Iptables targets

- j DROP

- j RETURN

- j REJECT
  - - reject- with
    - icmp- net- unreachable
    - icmp- host- unreachable
    - icmp- port- unreachable
    - ...
    - tcp- reset

Evasion

- Packet fragmentation
- Polymorphic shellcode
- URL encoding
- Session splicing (Whisker)

Snort_inline

- Inline IDS/IPS
- Linux bridge
- Netfilter libipq
- libnet

Snort_inline packet decisions

- Alert
- Drop
- Reject
- Replace

content: “/bin/sh”; replace: “/ben/sh”;

Snort_inline packet journey

- (kernel space) packet in ingress interface
- (kernel space) iptables FORWARD chain
- (kernel space) libipq
- (user space) context switch to Snort_inline
- (user space) Snort detection engine
- (user space) libnet
- (kernel space) packet on egress interface

Iptables string match patch

net/ipv4/netfilter/ipt_string.c

char * search(char *needle, char *haystack, int nlen, int hlen)

We get a pointer to the data!!!

Replace string interface

Snort SID 940: “WEB-FRONTPAGE shtml.dll

```
iptables -I FORWARD 1 -p tcp --dport 80 --
tcp-flags ACK ACK -m string  --string
“/_vti_bin/shtml.dll” --replace-string
“/vti_bin/shtml.doo” -j LOG --log-prefix
“nullify SID940 ”
```

Replace string interface (2)

Snort SID 261: “DNS EXPLOIT named overflow attempt”

iptables -I FORWARD 1 -p tcp --dport 53 --tcp-flags ACK ACK -m string --hex-string “|CD80 E8D7 FFFF FF|/bin/sh” --replace-hex-string “ben/sh” -j LOG --log-prefix “nullify SID261 ”

Iptables packet journey

- (kernel space) packet on ingress interface
- (kernel space) packet match in FORWARD chain
- (kernel space) string match function and data replacement
- (kernel space) packet on egress interface

Netperf bencharks

- Data port patch
- Example benchmarks. Linux-2.4.26 with string replace patch vs. Netperf and Apache-2.0.40

Applications?

- Well-defined exploits
- Preserve application layer responses

IPS

References

fwknop: http://www.cipherdyne.org/fwknop

fwsnort: http://www.cipherdyne.org/fwsnort

snort2iptables:
http://www.stearns.org/snort2iptables

Snort_inline: http://snort-inline.sourceforge.net

“The Base Rate Fallacy and its Implications for the Difficulty of Intrusion Detection”: http://www.raid-symposium.org/raid99/PAPERS/Axelsson.pdf