The Packets Need Me Do I?

* Getting started with distributed full packet capture using OpenFPC *

```plaintext
0x0000: 4518 005d de28 0000 2f11 b26d 0808 0808 E...\...m....
0x0010: c0a8 2a2a 0035 a4e8 0049 c7c9 ae3e 8180 \**.5...I...>
0x0020: 0001 0001 0000 0000 0664 6566 636f 6e03 ..............defcon.
0x0030: 6f72 6700 000f 0001 c00c 000f 0001 0000 org............
0x0040: 0000 0019 000a 066d 6169 6c2d 310a 6461 ........mail.l.da
0x0050: 7461 6d65 7269 6361 0363 6f6d 00 tamerica.com....
```
Session Overview

- Introductions
- What is Full Packet Capture
  - “Hey, that sounds like effort, why do it?”
- General implementation challenges
  - Operational considerations
  - Legal considerations
  - Other deployment options
- Introduction to OpenFPC
  - Use cases
  - Architecture
  - Lessons learned
- Live demo & tool overview
  - 'Wall of Sheep' use case
- How you can get Started
  - From zero to packets in a few commands
- How can you help?
A Quick Teaser:
The full packet

user@hostname:~/.bin$ sudo tcpdump -ni wlan1 -s0 -XX -c1 udp port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan1, link-type EN10MB (Ethernet), capture size 262144 bytes

0x0000: 94fe f478 8b0e c48e 8ff4 3fe5 0800 4500 .....x......?...E.
0x0010: 003c b0ef 4000 4011 b3f0 c0a8 2a7f c0a8 ..<..@.@.....*....
0x0020: 2a01 1515 0035 0028 c8a4 bc3c 0100 0001 *.....5.(....<.....
0x0030: 0000 0000 0000 0377 7777 0664 6566 636f ..........www.defcon
0x0040: 6e03 6f72 6700 0001 0001 n.org.....

1 packet captured
Simple eh?

• So why so hard?
  – Scale (volume & rate of data)
  – The distributed nature of networks
  – Accessing the data
  – Long (enough) term storage for human analysis

• Benefits
  – Verification
  – Great for skeptics
Scale

- Networks don't get slower

<table>
<thead>
<tr>
<th>Sustained Speed</th>
<th>Storage / Hour</th>
<th>Storage / day</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Mbps</td>
<td>4.9GB</td>
<td>106GB</td>
</tr>
<tr>
<td>100Mbps</td>
<td>43.9GB</td>
<td>~1TB</td>
</tr>
<tr>
<td>10Gbps</td>
<td>~4.5TB</td>
<td>~100TB</td>
</tr>
<tr>
<td>40Gbps</td>
<td>~18TB</td>
<td>~432TB</td>
</tr>
</tbody>
</table>

- Capture rate & load
  - Can load be distributed over multiple devices?

* All Super quick back of napkin math, doesn't cover CRC and other packet overheads. For illustrative purposes only
Distributed networks & data access

- Must be close to the data
- Asymmetric traffic is everywhere
  - Get over it...
- Use a central point for searching & extraction where possible
Good habits for security analysts
Packets or it didn't happen!

https://check.torproject.org/exit-addresses
ExitNode 0011BD2485AD45D984EC4159C88FC066E5E3300E
Published 2015-06-29 16:17:33
LastStatus 2015-06-30 11:02:31
ExitAddress 217.79.18.50 2015-07-28 11:10:15

user@hostname:~$ host pool.ntp.org
pool.ntp.org has address 217.79.18.50
pool.ntp.org has address x.x.x.x
pool.ntp.org has address x.x.x.x
pool.ntp.org has address x.x.x.x
Why don't all tools provide packets?

- Because “it's hard”
  - Market doesn't expect it
  - Resource impact (detection)
  - Storage
  - Can make products look complicated
  - “Just block it, problem solved”

- Speed wins, almost always
- Political/legal challenges
- Times are changing
Full Packet Capture

EASY, FREE, AND ON YOUR OWN HARDWARE

LEARN MORE
Alternative Options

- Moloch
  - github.com/aol/moloch
- OpenSOC
  - github.com/OpenSOC/opensoc
- PacketPig
  - github.com/packetloop/packetpig
- Sguil
  - bammv.github.io/sguil
- Stenographer
  - github.com/google/stenographer
Choosing where to focus

- Avoid
  - Vertical scaling
  - Threat detection

- Focus on
  - Decentralisation
  - Getting access to data
  - Automation
  - Simple integration
  - Time to get up and running
Ingredients

- **External**
  - Daemonlogger
  - Cxtracker & cx2db (cx2db forked to work with `<nodename>.conf`)
  - Mysql
  - Perl (Dancer)
- **Internal**
  - openfpc-queued
  - openfpc-client
  - openfpc-restapi
  - openfpc-ChromeExtension
  - `<nodename>.conf`
- **Misc helpers**
  - openfpc-passwd, openfpc-dbmaint, openfpc-install,
    openfpc, init scripts
Components

- **Server daemon: openfpc-queued**
  - Listens for network connections
  - Receives requests
    - Status, fetch, store, apikey, search
  - Executes requests and returns data back to the client
    - Integrated queueing system

- **Client software: openfpc-client**
  - Installed on analyst workstation
  - Perl with limited dependencies
    - Tested on OSX (perlbrew) & Ubuntu
High level diagram

- **Node**: Responsible for
  - Traffic capture
  - Session capture
  - Logging
  - Access Control
- **Platform**:
  - Linux
  - Ubuntu 14.04lts
Component level

Traffic for capture

Session file(s)

openfpc-cx2db

cstracker

daemonlogger

Session metadata

PCAP store

OpenFPC queued

RestAPI

Network Socket

OpenFPC- ChromeExtension

$ openfpc-client

OpenFPC Node ‘Paris’

/var/tmp/openfpc/pcaps/<NODE_NAME>.time
/var/tmp/openfpc/pcaps/<NODE_NAME>.time
/var/tmp/openfpc/pcaps/<NODE_NAME>.time

Extracted

/var/tmp/openfpc/extracted/R-GUID: (pcap, json, txt)
More complex configurations
Proxy Functionality

- Enables multiple devices function “as one”
- Removes need for central 'flow' / packet datastore
- “Stackable”

$ openfpc-client -a fetch -bpf “tcp port 31337 and host 1.1.1.1”

openfpc-client

--server JFK
--server US
--server World

OpenFPC Nodes

JFK
LAS
BWI

US

World

EU

LHR
ORY
AMS
OpenFPC Actions

• Fetch
  - Go get the packets, I'm waiting for them...
  - Multiple extractions can take place at once

• Search
  - Look for the sessions before getting the packets
  - Multiple execute at the same time

• Store
  - Add to the extract queue
  - Important for integration
A very British queuing system

```
$ openfpc-client -a store -dpt=53 -last 600

Queue Position : 3
Remote File    : EB08EA98-<SNIP>-66E2BD3.pcap
Result         : In Queue
```

```
$ curl https://<host>:port>...
{
"message":"In Queue",
"rid":"9B522-<snip>-2B67317",
"filename":"9B522-<snip>-2B67317.pcap",
"position":"3",
"md5": "0",
}
```
Reasoning behind the store action

- People are slow
  - Human analysis takes time, and you've not got enough of it
  - People need to sleep (but less so at Defcon)
Time as an anchor

- All requests need to be anchored to time somehow
  - 'stime/etime'
  - 'timestamp'
  - 'last <seconds>'

30/July/2015 11:00: Oldest packet

stime: 11:11 GMT
etime: 11:21 GMT
'host 1.1.1.1 and tcp port 25'

name.1438254000
name.1438254600
name.1438255120
name.1438255240
name.1438255340

openfpc-<nodename>.pcap.<timestamp>
Non-packet actions: Status

* Node: Test_Node

- Node Type                                      : NODE
- Description                                    : "Development Test Node"
- **Packet storage utilization**                 : 83 %
- Session storage utilization                    : 25 %
- Space available in save path                   : 25 %
- Space used in the save path                    : 11840216 (11.84 GB)
- Session storage used                           : 11840216 (11.84 GB)
- **Packet storage used**                        : 32391816 (32.39 GB)
- PCAP file space used                           : 31G
- Local time on node                             : 1436178831 (Mon Jul  6 11:33:51 2015 Europe/London)
- Oldest session in storage                      : 1435648639 (Tue Jun 30 09:22:59 2015 Europe/London)
- Oldest packet in storage                       : 1435648639 (Tue Jun 30 08:17:19 2015 Europe/London)
- **Storage Window**                             : 15 Days, 18 Hours, 10 Minutes, 29 Seconds
- Load Average 1                                 : 0.07
- Load average 5                                 : 0.03
- Load average 15                                : 0.05
- Number of session files lagging                : 0
- Number of sessions in Database                 : 5398178
- Node Timezone                                  : Europe/London
Constraints: Specifying data

$ openfpc-client -a fetch -last 600\
    sip=1.1.1.1 dip=2.2.2.2 dpt=80 proto=tcp

$ openfpc-client -a fetch -last 600 -bpf \\n    "host 1.1.1.1 and host 2.2.2.2 and tcp port 80"

$ openfpc-client -a fetch -logline \\
    "192.168.42.127,0,666,6,CLIENT,[http:Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36"


"**2010-04-05 10:23:12** 1NyiWV-0002\K-QJ <= loedgafe3\@nuttyeviltours.com H=(ABTS-AP-dynamic-111.119.119.122.airtelbroadband.in) [**111.119.119.122**] P=esmtp S=2056 id=000d01cad4a1\$ab5a3780\$6400a8c0\@fake3"
Tool → pcap → tool → pcap

Tool “x” identifies an interesting session, raises a log event

Extract packet data from OpenFPC using “logline x”

Tool “y” is used to inspect pcap and raises another log event

Extract packet data from OpenFPC using “logline y”
Example

“Is something suspicious going on with DNS from host 1.1.1.1?”

$ ./openfpc-client -a fetch -dpt 53 -sip 1.1.1.1 \ -w alldns.pcap

“Lets use a tool to look at that that pcap more closely...”

$ ./passivedns -r alldns.pcap
<snip>
1438092567||1.1.1.1||8.8.8.8||IN||evil.foo.com||A||2.2.2.2||54||13

“Hrmmm, need to take a look at that....”

$ ./openfpc-client -a fetch \ --logline “1438092567|1.1.1.1|8.8.8.8|IN|evil.foo.com|A|2.2.2.2|54|13”

“Right, so that’s what they were talking about...”
Thanks & help (please)

- Thanks to those that wrote key components and contributors:
  - Daemonlogger, Marty Roesch
  - Cxtracker, Edward Fjellskål

- Help:
  - Debian or Ubuntu upstream
  - CSS & Styling for OpenFPC-ChromeExtension
Demo time...

- Working through an OpenFPC Proxy
  - Session searching
  - And then grabbing the packets
- Wall of sheep use case
  - Credential extraction based on external detection
- Install
  - Zero to packets in 1 minute...