

Another solution to the Network Forensics Puzzle #3

publié par malphx le samedi, février 6 2010 - 11:13

For this [contest](#), We've had to deal with Apple TV traffic.

Tools Used:

_ TShark 1.2.2

_ [macfinder.rb](#) (custom ruby script giving IP/MAC bindings from a pcap file.) (require packetfu)

_ [httpdumper](#) (custom ruby script that can display and dump HTTP conversations) (require packetfu and terminal-table)

_ [plist.rb](#) (custom ruby script that can display informations extracted from Apple Property-List 1.0 XML documents)

Recommandations for using these tools:

macfinder and httpdumper rely on the wonderful ruby lib: [packetfu](#).

Significant performances improvements have been done by its author in version 0.3.1. (8x faster)

So, You **SHOULD** use the last version of packetfu, eg: at least, packetfu 0.3.1

.

Detailed AnswersAs usual, we have to verify the evidence file integrity:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ md5sum evidence03.pcap
```

```
f8a01fbe84ef960d7cbd793e0c52a6c9 evidence03.pcap
```

Ok, we're good to go !

First, I used to look at the protocol hierarchy stats given by tshark to take a first look at a pcap file:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ tshark -r evidence03.pcap
-qz io,phs
```

```
=====
```

```
Protocol Hierarchy Statistics
```

```
Filter: frame
```

```
frame frames:1778 bytes:1508750
```

```
eth frames:1778 bytes:1508750
```

```
ip frames:1778 bytes:1508750
```

```
udp frames:28 bytes:6102
```

```
dns frames:28 bytes:6102
```

```
tcp frames:1750 bytes:1502648
```

```
http frames:167 bytes:93189
```

```
image-gif frames:33 bytes:21202
xml frames:18 bytes:20852
tcp.segments frames:65 bytes:46469
http frames:65 bytes:46469
xml frames:17 bytes:11732
image-jfif frames:48 bytes:34737
=====
```

Hum, I bet we'll have to work with HTTP and some XML documents/data !

Let's continue...

We know that Ann has recently acquired an AppleTV and has configured it with a static IP address: 192.168.1.10

For some obvious reason, we have to know the mac (or hardware) address of Ann's new HDTV box.

Tshark could help for this task:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ tshark -r evidence03.pcap
-R "ip.src==192.168.1.10" -Tfields -e "eth.src" |uniq
00:25:00:fe:07:c4
```

I've also coded a small ruby script for this task: macfinder.rb. Here's the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb
```

```
macfinder version 0.1
Copyright © 2009 Franck GUENICHOT
macfinder comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome
to redistribute it under certain conditions.
(GPL v3)

Usage: macfinder [options]
-i, -ip Display Mac address for the given IP address only
(4-digit decimal dot notation form)
-v, -version Display version information
-h, -help Display this screen
```

Without any switch, macfinder.rb displays all the source IP/MAC address found in the pcap file:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb
evidence03.pcap
```

Listing all Mac Address found !

```
IP: 8.18.65.10 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.32 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.88 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.89 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.67 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.58 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.82 | Mac: 00:23:69:ad:57:7b
IP: 8.18.65.27 | Mac: 00:23:69:ad:57:7b
IP: 192.168.1.10 | Mac: 00:25:00:fe:07:c4
IP: 4.2.2.1 | Mac: 00:23:69:ad:57:7b
IP: 66.235.132.121 | Mac: 00:23:69:ad:57:7b
```

From the listing above we can easily find Ann's AppleTV mac address.

But to be less verbose, and because we know the IP address, we can use the -i switch to display only the interesting MAC.

Here's the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./macfinder.rb -i
192.168.1.10 evidence03.pcap
Mac: 00:25:00:fe:07:c4
```

And Voila !

Going Deeper Part I : HTTP

Now, we have to go deeper in the pcap file to analyse Ann's networking activity and particularly

her AppleTV network conversations.

Tshark let us know that we'll have to deal with HTTP (and maybe XML documents, later), so I wrote a specialized tools

to facilitate the investigation: httpdumper

httpdumper basically displays informations about HTTP conversations. The HTTP protocol is a Request/Response protocol meaning

that a client makes a request to a server with HTTP request messages and the server answers with HTTP response messages.

httpdumper handles this mechanism and displays these conversations in an easy to understand manner.

Some terminology:

An HTTP conversation, for httpdumper, is the set of all REQUEST/RESPONSE HTTP messages involving the same 2 hosts and tcp ports.

An HTTP flow is an unidirectionnal flow of http data (eg: client to server (request) or server to client (response))

Here's the help screen:

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./httpdumper -h
```

```
httpdumper version 0.1
Copyright © 2010 Franck GUENICHOT
httpdumper comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome
to redistribute it under certain conditions.
(GPL v3)
```

```
Usage: httpdumper [options] -r
-r, -read Read the given pcap file [REQUIRED]
-c, -conversation # List only flows for conversation #
-f, -flow # List only flow #
-with-headers For Display ONLY
-d, -dump Dump the selected conversation or flow
-p, -port Define custom HTTP port
-s, -stats type,[val1],[val2] Displays statistics
Valid options:
Request stats: request,[requester_ip],[requested_host]
URI list: uri,[requester_ip],[target_hostname]
-v, -version Display version information
-h, -help Display this screen
```

httpdumper is only a passive (lightweight) analysis tool, it needs a file in entry, so -r options is required to launch this tool.

The default output (without any options) displays all the HTTP conversations found in the given pcap file.

Let's do it !

```
franck@ODIN:~/Analysis/Sources/Puzzle_3$ ./httpdumper -r
evidence03.pcap
Reading file evidence03.pcap
Parsing packets...
1778 packets read in 4.385 sec.
```

Found 20 HTTP conversation(s)

Conversation Index	Hosts	HTTP Flow count	Request	Response
0	192.168.1.10:49163 8.18.65.67:80	2	1	1
1	192.168.1.10:49164 66.235.132.121:80	2	1	1
2	192.168.1.10:49165 8.18.65.32:80	8	4	4
3	192.168.1.10:49166 66.235.132.121:80	8	4	4
4	192.168.1.10:49167 8.18.65.58:80	20	10	10
5	192.168.1.10:49168 8.18.65.67:80	4	2	2
6	192.168.1.10:49169 66.235.132.121:80	2	1	1

7	192.168.1.10:49170	8.18.65.82:80	44	22	22	675124	
8	192.168.1.10:49171	8.18.65.27:80	6	3	3	13582	
9	192.168.1.10:49172	66.235.132.121:80	6	3	3	129	
10	192.168.1.10:49173	8.18.65.27:80	8	4	4	12744	
11	192.168.1.10:49174	66.235.132.121:80	2	1	1	43	
12	192.168.1.10:49175	66.235.132.121:80	6	3	3	129	
13	192.168.1.10:49176	8.18.65.67:80	4	2	2	3493	
14	192.168.1.10:49177	8.18.65.10:80	32	16	16	362826	
15	192.168.1.10:49178	66.235.132.121:80	2	1	1	43	
16	192.168.1.10:49179	8.18.65.88:80	20	10	10	5576	
17	192.168.1.10:49180	66.235.132.121:80	20	10	10	430	
18	192.168.1.10:49181	8.18.65.89:80	18	9	9	4861	
19	192.168.1.10:49182	66.235.132.121:80	18	9	9	387	
+-----+-----+-----+-----+-----+-----+-----+-----+							

The table above show all the HTTP conversations found. (this kind of table is best viewed on large display)

The flow count indicates the number of flows in each conversations

Request and Response column, each displays the number of HTTP Request or HTTP response in each conversation

Cumulative length: the length (in Bytes) of the HTTP Payloads (or HTTP message body) exchanged in each conversation.

Note: this length takes only HTTP Response payloads into account. By now HTTP Request message body, if any, is not displayed (and not dumpable")

Quickly, we learn interesting infos:

- _ 20 HTTP conversations, all involving the same client (Ann's AppleTV)
- _ 7 of them are composed of 18+ flows
- _ Conversation #7 has the greater cumulative length (so, the largest HTTP payload)