SharkFest'17 US

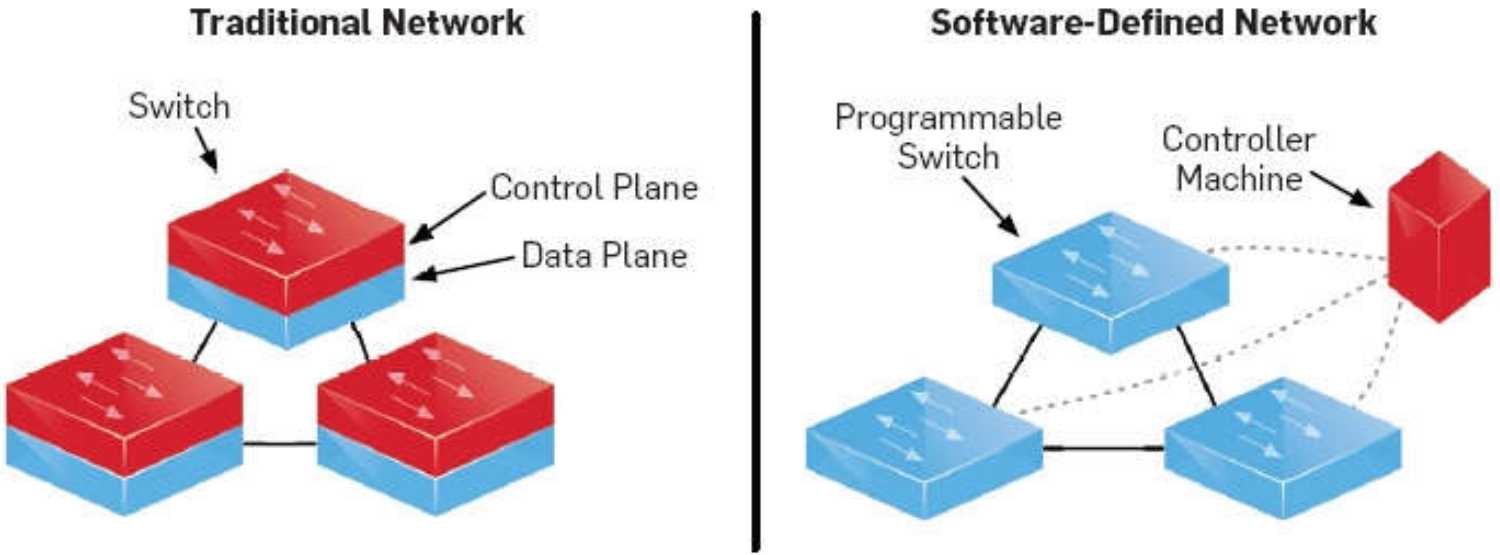
How tshark saved my SDN Forensics

Joseph Bull & Mike McAlister
Booz Allen Hamilton

SharkFest'17 US • Carnegie Mellon University • June 19-22, 2017

Background

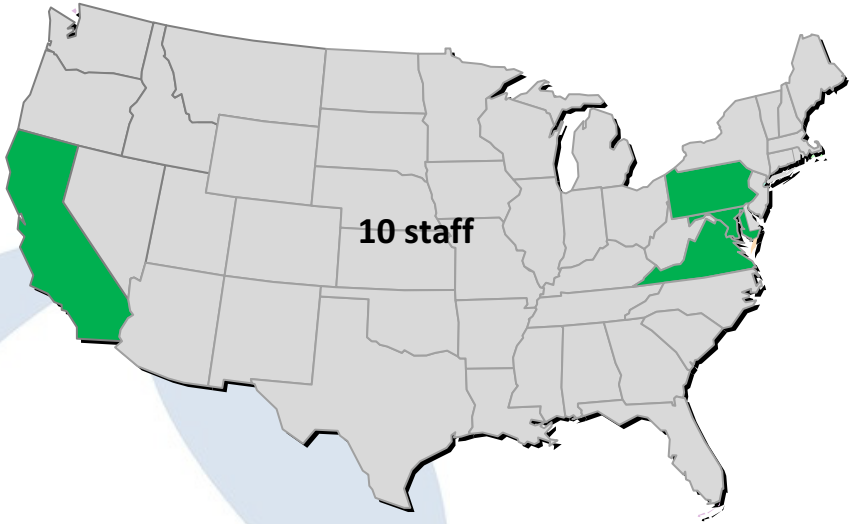
- Centralized control of network devices (SDN Controller) rather than need to leverage command line interface (CLI) for each device
- Transition to open source software/whitebox hardware removes vendor lock-in
- Introduction of new protocols (e.g., Openflow)
- Increased scalability from inherent automation and centralized control



- Digital Forensic Research Workshop (DFRWS) - the 2016 Forensic Challenge was specific to Software Defined Networking (SDN)
- Provided a PCAP file of the southbound traffic (between the SDN controller and virtual switch) (<2s traffic) & memory dump of the virtual switch. No other knowledge or clues of network setup were provided.
- Goals are to determine
  - Type of controller and switch
  - Which hosts were connected to which switch ports
  - How much traffic was sent by each host
  - Details about flow rules
  - Required development of **automated tool(s)** to complete forensics

Booz Allen won the International SDN Forensics Challenge

Geographically Diverse



Skill Mix

- Protocol Analysts
- Network Engineers (SDN & Openflow)
- Memory Analysts / Reverse Engineers
- Scripters

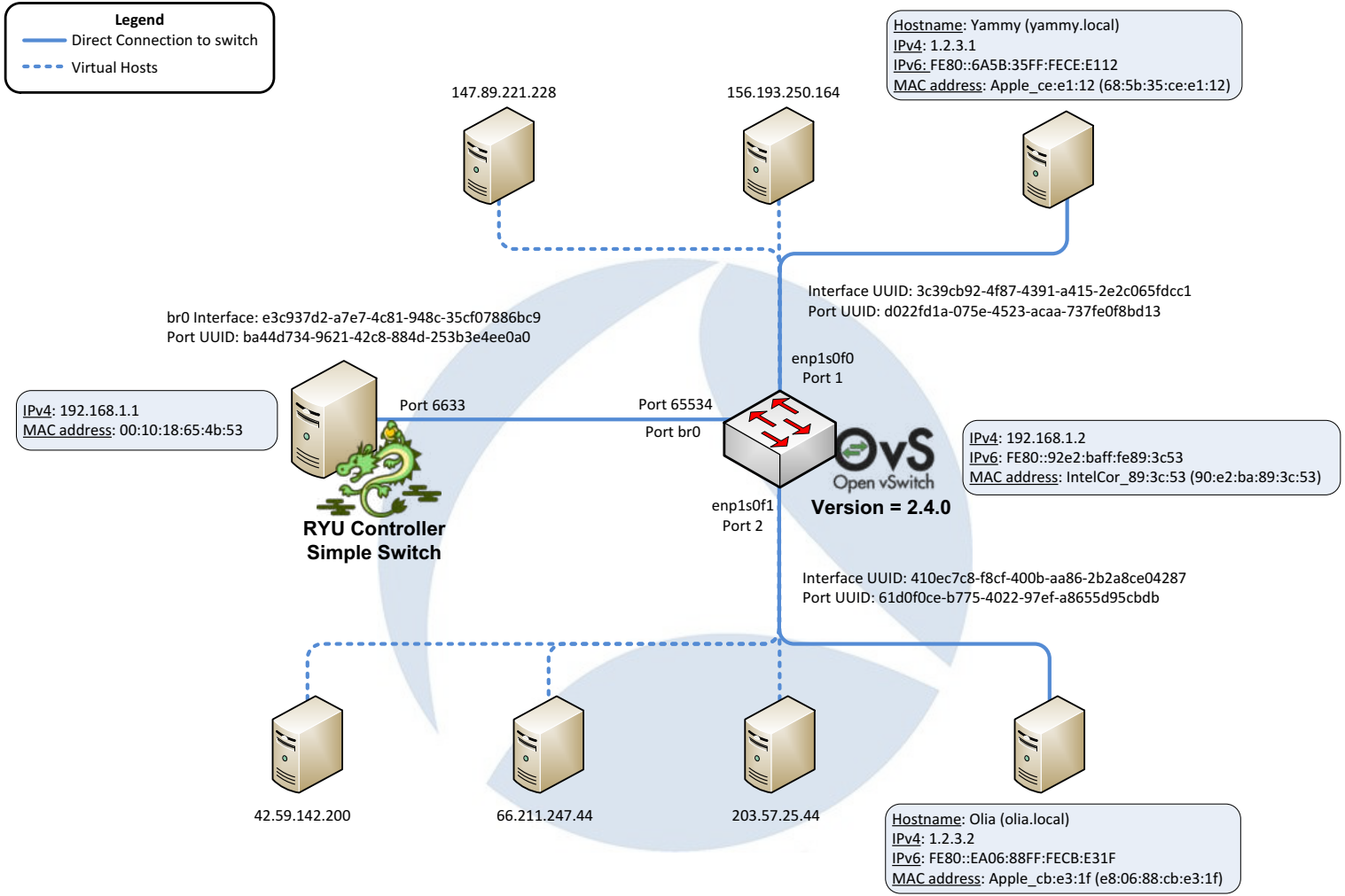
- Wireshark / tshark → used to assess the PCAP file including several important dissectors that supported the forensic analysis (OpenFlow, SSL)

- Volatility → enabled searching the memory for artifacts and reconstructing the file system of the memory provided

- Python → programming language that easily integrated with Wireshark and Volatility to perform the forensics necessary; additional criteria: also requires Pandas module and dependencies

Results

Question	Finding
Controller	RYU
Switch	OpenvSwitch version 2.4.0
Hosts	Two hosts directly connected to the switch and five other hosts were reachable via specific ports
Packets	• 16 packets (1553 bytes) for one flow rule (in_port=2, dl_dst=68:5b:35:ce:e1:12, action=output:1) • 14 packets (1360 bytes) for another (flow rule: in_port=1, dl_dst=e8:06:88:cb:e3:1f)
Flow rules	Reconstructed entire flow table by analyzing OpenFlow messages within PCAP - seven flow rules identified
Dynamic rules	Identified two dynamically set flows, each had a hard-timeout
Flow rule actions	• Output to switch port (OFPAT_OUTPUT) • Setting the 802.1q VLAN id (OFPAT_SET_VLAN_VID) • Setting the Ethernet source address (OFPAT_SET_DL_SRC) • Setting the IP destination address (OSPFAT_SET_NW_DST)


Legend
Direct Connection to switch
Virtual Hosts
Hostname: Yammy (yammy.local)
IPv4: 1.2.3.1
IPv6: FE80::6A5B:35FF:FECE:E112
MAC address: Apple_ce:e1:12 (68:5b:35:ce:e1:12)
147.89.221.228
156.193.250.164
Interface UUID: 3c39cb92-4f87-4391-a415-2e2c065fdcc1
Port UUID: d022fd1a-075e-4523-acaa-737fe0f8bd13
br0 Interface: e3c937d2-a7e7-4c81-948c-35cf07886bc9
Port UUID: ba44d734-9621-42c8-884d-253b3e4ee0a0
enp1s0f0
Port 1
IPv4: 192.168.1.1
MAC address: 00:10:18:65:4b:53
Port 6633
Port 65534
Port br0
OvS
Open vSwitch
Version = 2.4.0
IPv4: 192.168.1.2
IPv6: FE80::92e2:baff:fe89:3c53
MAC address: IntelCor_89:3c:53 (90:e2:ba:89:3c:53)
RYU Controller
Simple Switch
enp1s0f1
Port 2
Interface UUID: 410ec7c8-f8cf-400b-aa86-2b2a8ce04287
Port UUID: 61d0f0ce-b775-4022-97ef-a8655d95cbdb
42.59.142.200
66.211.247.44
203.57.25.44
Hostname: Olia (olia.local)
IPv4: 1.2.3.2
IPv6: FE80::EA06:88FF:FECB:E31F
MAC address: Apple_cb:e3:1f (e8:06:88:cb:e3:1f)

Additional Details	Finding
Log entries extracted	3306
OS of Virtual switch	LinuxFedora22-4_2_6-200x64
Details certs found within memory	(/C=US/ST=CA/O=Open vSwitch/OU=controllerca/CN=OVS controllerca CA Certificate (2015 Nov 24 12:51:33)) (fingerprint 27:6c:d9:23:3c:89:70:5a:01:cb:c7:7c:d6:bd:83:76:52:f9:95:44)
Git repositories installed	• git://github.com/osrg/ryu.git, • https://github.com/504ensicsLabs/LiME.git)
Several memory dumps were taken	/home/ram-base2.raw; /home/ram.raw; /home/ram2.raw; /home/ram3.raw
vSwitch capabilities	• Flow statistics = TRUE • Table statistics = TRUE • Port statistics = TRUE • Queue statistics = TRUE • Switch will block looping ports = FALSE

Frame Number	OFP_FRAME_TYPE	IN_PORT	DL_VLAN	DL_SRC	DL_DST	DL_TYPE	NW_PROTO	TP_SRC	TP_DST	SRC_WLDCD	DST_WLDCD	VLAN_PCP	NW_TOS	IN_PORT2	Eth Src
22	(14)OFPT_FLOW_MOD	IN_PORT	*	*	*	DL_TYPE	*	DL_SRC	DL_DST	2	60	*	*	1	00:00:00:00:00:00
24	(14)OFPT_FLOW_MOD	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	60	*	*	2	00:00:00:00:00:00
24	(14)OFPT_FLOW_MOD	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	60	*	*	2	00:00:00:00:00:00
24	(14)OFPT_FLOW_MOD	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	60	*	*	2	00:00:00:00:00:00
24	(14)OFPT_FLOW_MOD	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	60	*	*	1	00:00:00:00:00:00
63	(14)OFPT_FLOW_MOD	IN_PORT	*	*	DL_DST	*	*	*	*	63	63	*	*	1	00:00:00:00:00:00
67	(14)OFPT_FLOW_MOD	IN_PORT	*	*	DL_DST	*	*	*	*	63	63	*	*	2	00:00:00:00:00:00
197	(11)OFPT_FLOW_REMOVED	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	0	*	*	1	00:00:00:00:00:00
216	(11)OFPT_FLOW_REMOVED	IN_PORT	*	*	*	DL_TYPE	*	*	*	2	0	*	*	2	00:00:00:00:00:00

Frame Number	openflow.eth_dst	In VLAN ID	In VLAN Pri	IP ToS	IP Protocol	Src IP	Dest IP	Src Port	Dest Port	Command	Idle-Timeout	Hard-Timeout	Priority
22	00:00:00:00:00:00	0	0	0	0	147.89.221.228	0.0.0.0	80	99	New Flow	30	30	32768
24	00:00:00:00:00:00	0	0	0	0	42.59.142.200	0.0.0.0	0	0	New Flow	0	0	32768
24	00:00:00:00:00:00	0	0	0	0	66.211.247.44	0.0.0.0	0	0	New Flow	45	45	32768
24	00:00:00:00:00:00	0	0	0	0	203.57.25.44	0.0.0.0	0	0	New Flow	0	0	32768
24	00:00:00:00:00:00	0	0	0	0	156.193.250.164	0.0.0.0	0	0	New Flow	0	0	32768
63	Apple_cb:e3:1f(e8:06:88:cb:e3:1f)	0	0	0	0	0.0.0.0	0.0.0.0	0	0	New Flow	0	0	32768
67	Apple_ce:e1:12(68:5b:35:ce:e1:12)	0	0	0	0	0.0.0.0	0.0.0.0	0	0	New Flow	0	0	32768
197	00:00:00:00:00:00	0	0	0	0	93:59:DD:E4 (147.89.221.228)	0.0.0.0	0	0	N/A	N/A	N/A	N/A
216	00:00:00:00:00:00	0	0	0	0	42:D3:F7:2C (66.211.247.44)	0.0.0.0	0	0	N/A	N/A	N/A	N/A

Frame Number	Outport	Flags	action_header-type	action_header-value	Priority2	Reason	duration_sec	duration_nsec	idle_timeout	packet count	byte count
22	65535	1(SEND_FLOW_REM)	(7)OFPAT_SET_NW_DST	(0x63 66 2c 48) (99.102.44.72)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
24	65535	1(SEND_FLOW_REM)	(4)OFPAT_SET_DL_SRC	(41:31:3a:38:42:3a)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
24	65535	1(SEND_FLOW_REM)	(5)OFPAT_SET_DL_DST	(35:46:3a:45:37:3a)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
24	65535	1(SEND_FLOW_REM)	(1)OFPAT_SET_VLAN_VID	(0x01 02) (258)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
24	65535	1(SEND_FLOW_REM)	(2)OFPAT_SET_VLAN_PCP	(0x02)(2)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
63	65535	1(SEND_FLOW_REM)	(0)OFPAT_OUTPUT	(0x00 02)(Port 2) (0xff e5) (MaxLen - 65509)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
67	65535	1(SEND_FLOW_REM)	(0)OFPAT_OUTPUT	(0x00 01)(Port 1) (0xff e5) (MaxLen - 65509)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
197	N/A	N/A	N/A	N/A	32768	1(OFPRR_HARD_TIMEOUT)	00001e (30)	04:e3:38:80(82000000)	00 1e (30)	0	0
216	N/A	N/A	N/A	N/A	32768	1(OFPRR_HARD_TIMEOUT)	00002d(45)	05:A9:95:C0 (95000000)	00 2d (45)	0	0

Forensic Steps

- Four major forensic activities were required to answer the challenge questions posed by DFRWS:

    1. Integration with Volatility
    2. Recovery of file system and other memory details
    3. **Decryption of SSL/TLS traffic**
    4. **PCAP assessment**

- These steps were then further refined and automated through the development of python scripts that integrated Volatility and tshark

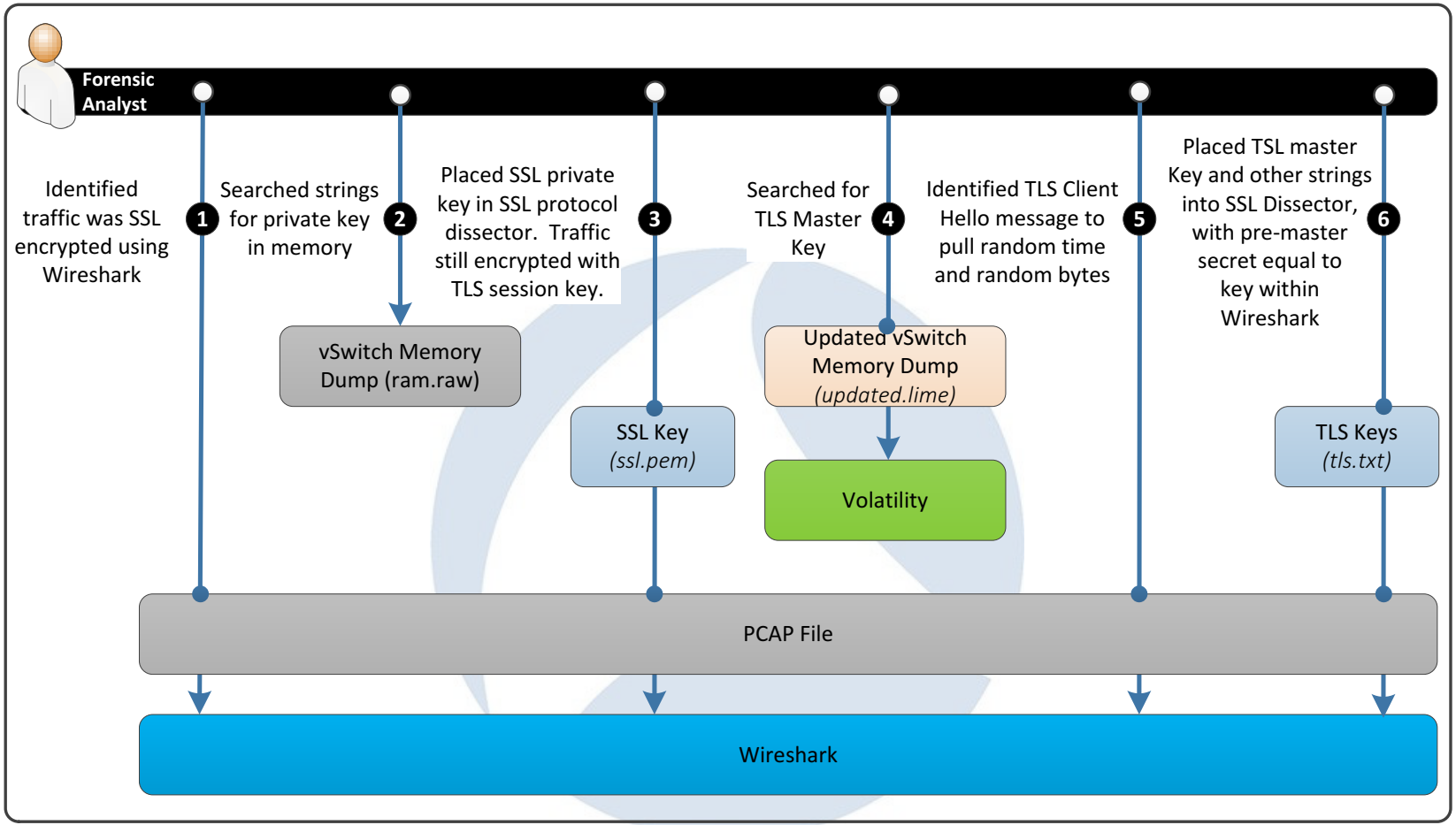
- WireShark
  - Supported our initial forensic analysis through visual review / filtering
  - Great to use for prototyping what will be at the command line
  - Did not allow us to automate our forensic solution (was not intended to)
  - Helped set preferences and ssl_keys file which is sourced by tshark

- tshark
  - Enabled automation of various steps of our forensics work
  - Easier to loop through and controlled outputs directly
  - SUPER friendly to grep, sed, awk, and other amazing Unix/Linux commands!

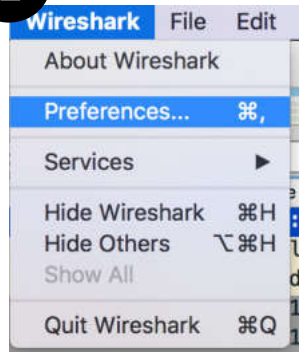
Application of tshark

SharkFest'17 US • Carnegie Mellon University • June 19-22, 2017

- Philosophy
- How we solved it manually in Wireshark
- How we automated it

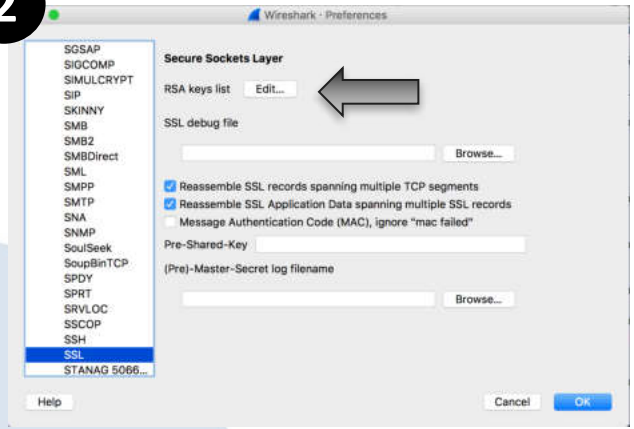

Forensic Analyst
Identified traffic was SSL encrypted using Wireshark
1
Searched strings for private key in memory
2
Placed SSL private key in SSL protocol dissector. Traffic still encrypted with TLS session key.
3
Searched for TLS Master Key
4
Identified TLS Client Hello message to pull random time and random bytes
5
Placed TSL master Key and other strings into SSL Dissector, with pre-master secret equal to key within Wireshark
6
vSwitch Memory Dump (ram.raw)
SSL Key (ssl.pem)
Updated vSwitch Memory Dump (updated.lime)
Volatility
TLS Keys (tls.txt)
PCAP File
Wireshark

1



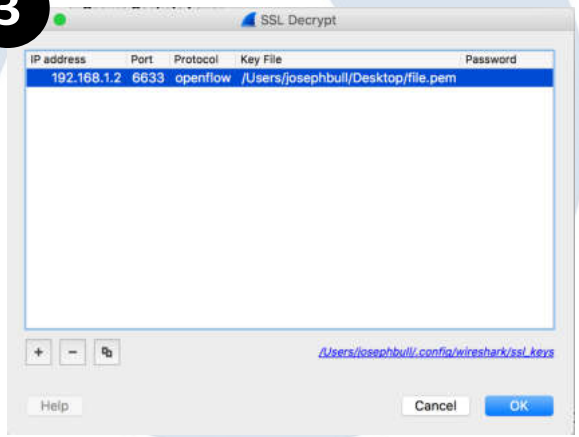
- Click on Preferences

2



- Expand Protocols
- Click on SSL
- Click Edit next to ‘RSA keys list’

3



- Click on the “+” to add a key
- Enter in details (see inputs)
- Select key file: Desktop/SDN Training/rsa_p_key.pem
- Select OK
- Select OK

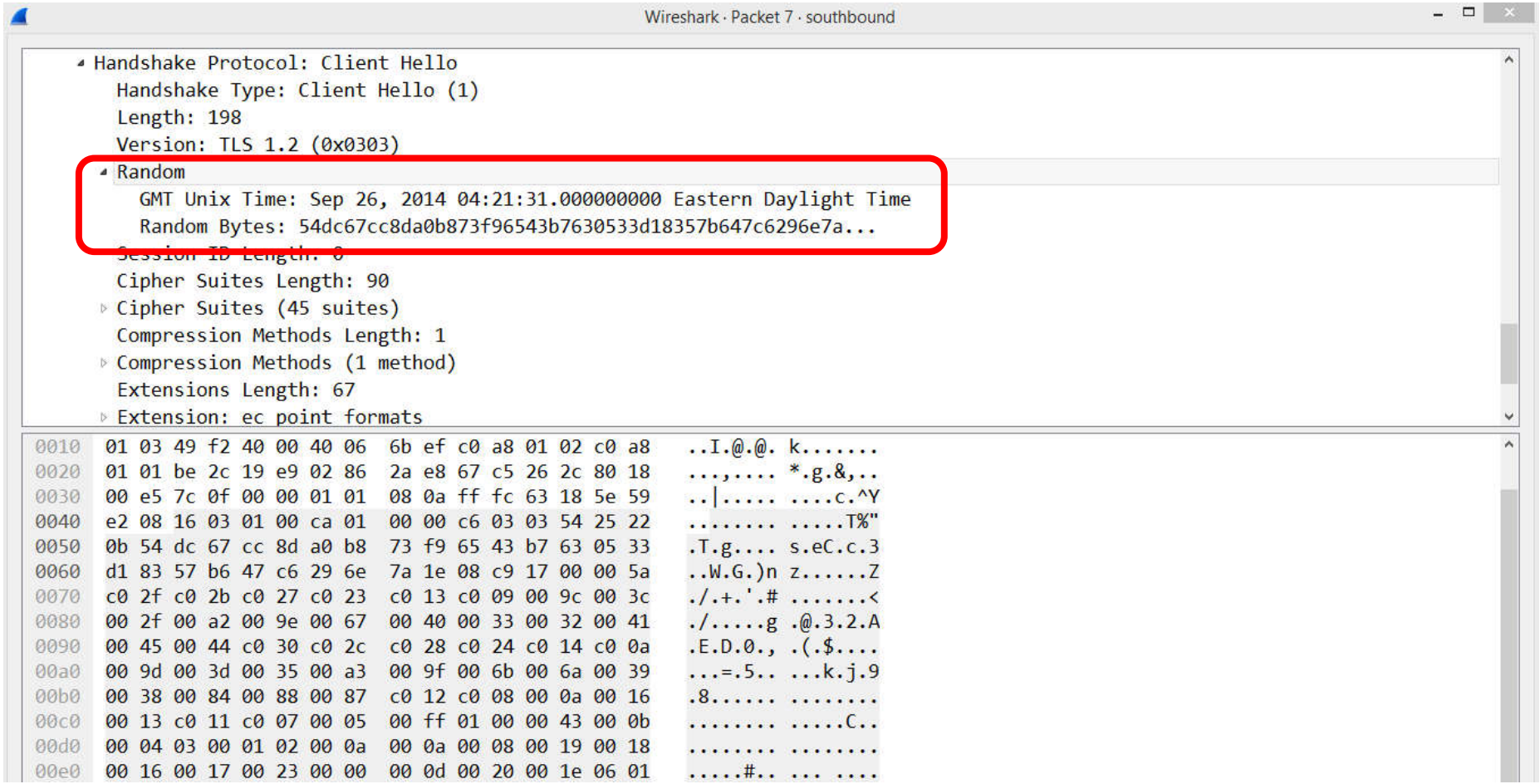
IP = 192.168.1.2
Port = 6633
Protocol = openflow
File = rsa_p_key.pem
Password = N/A

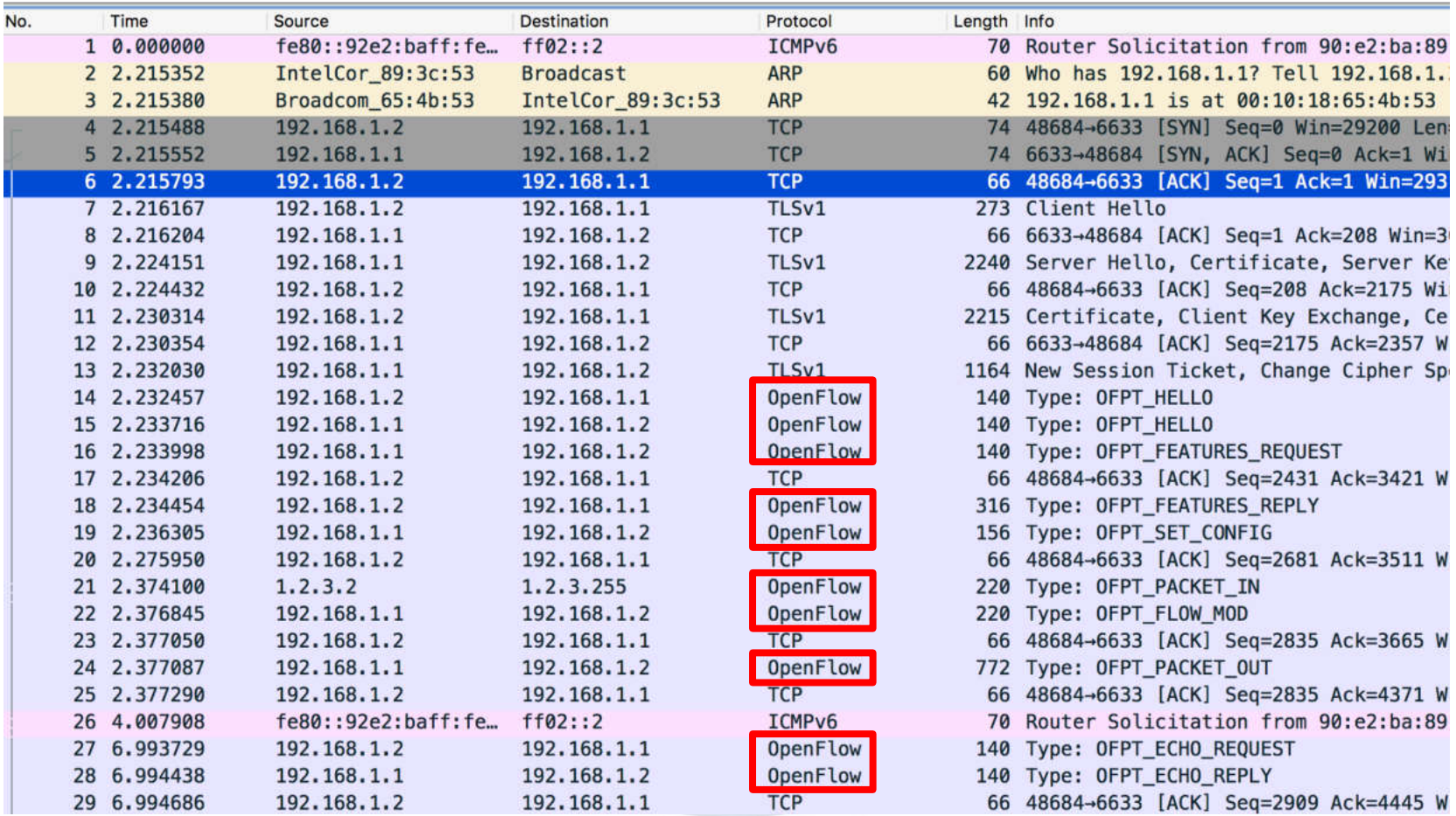
Encrypted PCAP

	Time	Source	Destination	Protocol	Length
1	2015/…	fe80::92e2:…	ff02::2	ICMPv6	70
2	2015/…	IntelCor_89…	Broadcast	ARP	60
3	2015/…	Broadcom_65…	IntelCor_89…	ARP	42
4	2015/…	192.168.1.2	192.168.1.1	TCP	74
5	2015/…	192.168.1.1	192.168.1.2	TCP	74
6	2015/…	192.168.1.2	192.168.1.1	TCP	66
7	2015/…	192.168.1.2	192.168.1.1	TCP	273
8	2015/…	192.168.1.1	192.168.1.2	TCP	66
9	2015/…	192.168.1.1	192.168.1.2	OpenFlow	2240
10	2015/…	192.168.1.2	192.168.1.1	TCP	66
11	2015/…	192.168.1.2	192.168.1.1	OpenFlow	2215
12	2015/…	192.168.1.1	192.168.1.2	TCP	66
13	2015/…	192.168.1.1	192.168.1.2	TCP	1164
14	2015/…	192.168.1.2	192.168.1.1	TCP	140
15	2015/…	192.168.1.1	192.168.1.2	TCP	140
16	2015/…	192.168.1.1	192.168.1.2	TCP	140
17	2015/…	192.168.1.2	192.168.1.1	TCP	66
18	2015/…	192.168.1.2	192.168.1.1	TCP	316
19	2015/…	192.168.1.1	192.168.1.2	TCP	156
20	2015/…	192.168.1.2	192.168.1.1	TCP	66
21	2015/…	192.168.1.2	192.168.1.1	TCP	220
22	2015/…	192.168.1.1	192.168.1.2	TCP	220
23	2015/…	192.168.1.2	192.168.1.1	TCP	66
24	2015/…	192.168.1.1	192.168.1.2	TCP	772

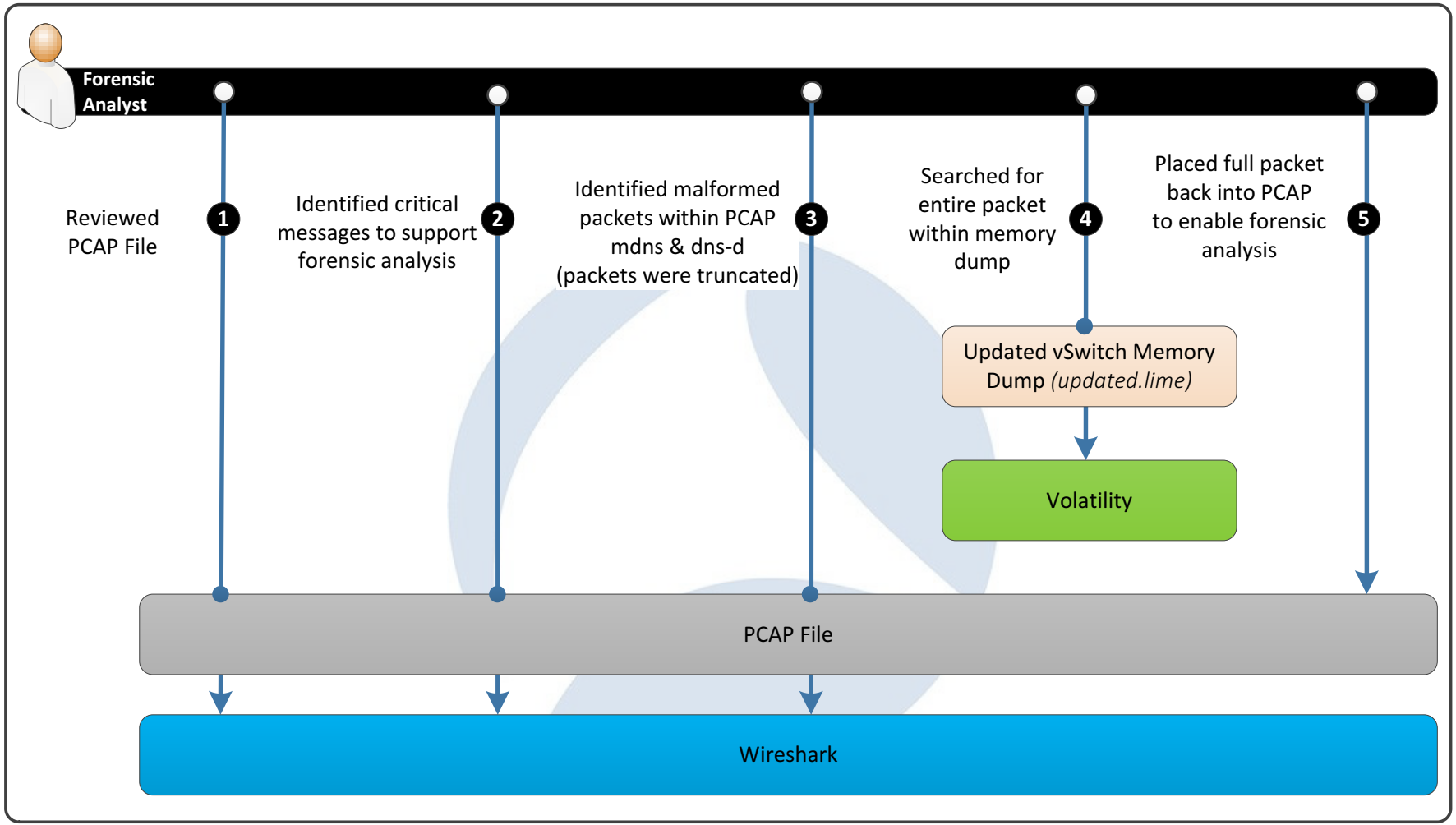
SSL Decrypted

	Time	Source	Destination	Protocol	Length
1	2015/…	fe80::92e2:…	ff02::2	ICMPv6	70
2	2015/…	IntelCor_89…	Broadcast	ARP	60
3	2015/…	Broadcom_65…	IntelCor_89…	ARP	42
4	2015/…	192.168.1.2	192.168.1.1	TCP	74
5	2015/…	192.168.1.1	192.168.1.2	TCP	74
6	2015/…	192.168.1.2	192.168.1.1	TCP	66
7	2015/…	192.168.1.2	192.168.1.1	TLSv1	273
8	2015/…	192.168.1.1	192.168.1.2	TCP	66
9	2015/…	192.168.1.1	192.168.1.2	TLSv1	2240
10	2015/…	192.168.1.2	192.168.1.1	TCP	66
11	2015/…	192.168.1.2	192.168.1.1	TLSv1	2215
12	2015/…	192.168.1.1	192.168.1.2	TCP	66
13	2015/…	192.168.1.1	192.168.1.2	TLSv1	1164
14	2015/…	192.168.1.2	192.168.1.1	TLSv1	140
15	2015/…	192.168.1.1	192.168.1.2	TLSv1	140
16	2015/…	192.168.1.1	192.168.1.2	TLSv1	140
17	2015/…	192.168.1.2	192.168.1.1	TCP	66
18	2015/…	192.168.1.2	192.168.1.1	TLSv1	316
19	2015/…	192.168.1.1	192.168.1.2	TLSv1	156
20	2015/…	192.168.1.2	192.168.1.1	TCP	66
21	2015/…	192.168.1.2	192.168.1.1	TLSv1	220
22	2015/…	192.168.1.1	192.168.1.2	TLSv1	220
23	2015/…	192.168.1.2	192.168.1.1	TCP	66
24	2015/…	192.168.1.1	192.168.1.2	TLSv1	772


Wireshark · Packet 7 · southbound
Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 198
Version: TLS 1.2 (0x0303)
Random
GMT Unix Time: Sep 26, 2014 04:21:31.000000000 Eastern Daylight Time
Random Bytes: 54dc67cc8da0b873f96543b7630533d18357b647c6296e7a...
Session ID Length: 0
Cipher Suites Length: 90
Cipher Suites (45 suites)
Compression Methods Length: 1
Compression Methods (1 method)
Extensions Length: 67
Extension: ec point formats
0010 01 03 49 f2 40 00 40 06 6b ef c0 a8 01 02 c0 a8 ..I.@.@. k.....
0020 01 01 be 2c 19 e9 02 86 2a e8 67 c5 26 2c 80 18 ...,.... *.g.&,..
0030 00 e5 7c 0f 00 00 01 01 08 0a ff fc 63 18 5e 59 ..|..... ....c.^Y
0040 e2 08 16 03 01 00 ca 01 00 00 c6 03 03 54 25 22 ....T%"
0050 0b 54 dc 67 cc 8d a0 b8 73 f9 65 43 b7 63 05 33 .T.g.... s.eC.c.3
0060 d1 83 57 b6 47 c6 29 6e 7a 1e 08 c9 17 00 00 5a ..W.G.)n z.....Z
0070 c0 2f c0 2b c0 27 c0 23 c0 13 c0 09 00 9c 00 3c ./.+.'.# .....<
0080 00 2f 00 a2 00 9e 00 67 00 40 00 33 00 32 00 41 ./.....g .@.3.2.A
0090 00 45 00 44 c0 30 c0 2c c0 28 c0 24 c0 14 c0 0a .E.D.0., .($....
00a0 00 9d 00 3d 00 35 00 a3 00 9f 00 6b 00 6a 00 39 ...=.5.. ...k.j.9
00b0 00 38 00 84 00 88 00 87 c0 12 c0 08 00 0a 00 16 .8..... ....
00c0 00 13 c0 11 c0 07 00 05 00 ff 01 00 00 43 00 0b ....C..
00d0 00 04 03 00 01 02 00 0a 00 0a 00 08 00 19 00 18 ....
00e0 00 16 00 17 00 23 00 00 00 0d 00 20 00 1e 06 01 .....#.. ... .



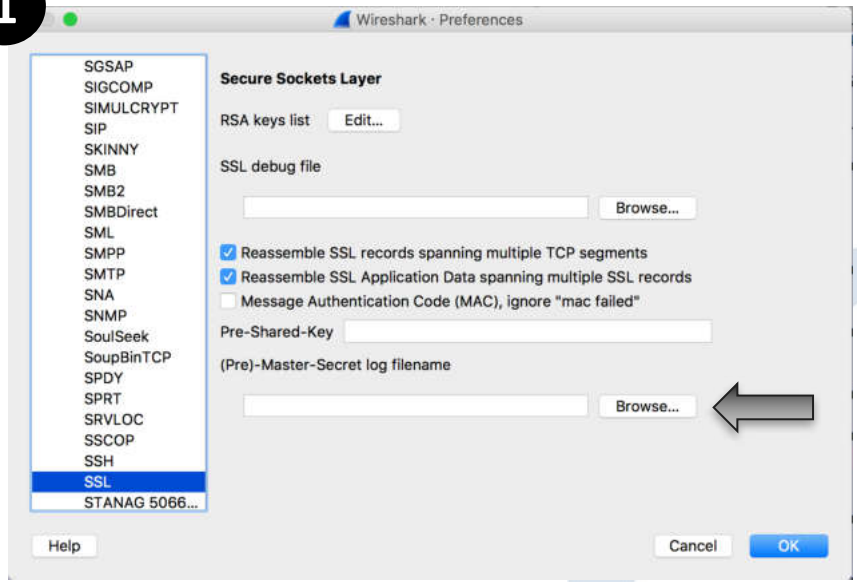
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::92e2:baff:fe…	ff02::2	ICMPv6	70	Router Solicitation from 90:e2:ba:89
2	2.215352	IntelCor_89:3c:53	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.
3	2.215380	Broadcom_65:4b:53	IntelCor_89:3c:53	ARP	42	192.168.1.1 is at 00:10:18:65:4b:53
4	2.215488	192.168.1.2	192.168.1.1	TCP	74	48684→6633 [SYN] Seq=0 Win=29200 Len
5	2.215552	192.168.1.1	192.168.1.2	TCP	74	6633→48684 [SYN, ACK] Seq=0 Ack=1 Wi
6	2.215793	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=1 Ack=1 Win=293
7	2.216167	192.168.1.2	192.168.1.1	TLSv1	273	Client Hello
8	2.216204	192.168.1.1	192.168.1.2	TCP	66	6633→48684 [ACK] Seq=1 Ack=208 Win=3
9	2.224151	192.168.1.1	192.168.1.2	TLSv1	2240	Server Hello, Certificate, Server Ke
10	2.224432	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=208 Ack=2175 Wi
11	2.230314	192.168.1.2	192.168.1.1	TLSv1	2215	Certificate, Client Key Exchange, Ce
12	2.230354	192.168.1.1	192.168.1.2	TCP	66	6633→48684 [ACK] Seq=2175 Ack=2357 W
13	2.232030	192.168.1.1	192.168.1.2	TLSv1	1164	New Session Ticket, Change Cipher Sp
14	2.232457	192.168.1.2	192.168.1.1	OpenFlow	140	Type: OFPT_HELLO
15	2.233716	192.168.1.1	192.168.1.2	OpenFlow	140	Type: OFPT_HELLO
16	2.233998	192.168.1.1	192.168.1.2	OpenFlow	140	Type: OFPT_FEATURES_REQUEST
17	2.234206	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=2431 Ack=3421 W
18	2.234454	192.168.1.2	192.168.1.1	OpenFlow	316	Type: OFPT_FEATURES_REPLY
19	2.236305	192.168.1.1	192.168.1.2	OpenFlow	156	Type: OFPT_SET_CONFIG
20	2.275950	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=2681 Ack=3511 W
21	2.374100	1.2.3.2	1.2.3.255	OpenFlow	220	Type: OFPT_PACKET_IN
22	2.376845	192.168.1.1	192.168.1.2	OpenFlow	220	Type: OFPT_FLOW_MOD
23	2.377050	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=2835 Ack=3665 W
24	2.377087	192.168.1.1	192.168.1.2	OpenFlow	772	Type: OFPT_PACKET_OUT
25	2.377290	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=2835 Ack=4371 W
26	4.007908	fe80::92e2:baff:fe…	ff02::2	ICMPv6	70	Router Solicitation from 90:e2:ba:89
27	6.993729	192.168.1.2	192.168.1.1	OpenFlow	140	Type: OFPT_ECHO_REQUEST
28	6.994438	192.168.1.1	192.168.1.2	OpenFlow	140	Type: OFPT_ECHO_REPLY
29	6.994686	192.168.1.2	192.168.1.1	TCP	66	48684→6633 [ACK] Seq=2909 Ack=4445 W


Forensic Analyst
Reviewed PCAP File
1
Identified critical messages to support forensic analysis
2
Identified malformed packets within PCAP mdns & dns-d (packets were truncated)
3
Searched for entire packet within memory dump
4
Placed full packet back into PCAP to enable forensic analysis
5
Updated vSwitch Memory Dump (updated.lime)
Volatility
PCAP File
Wireshark

Wireshark Time!

SharkFest'17 US • Carnegie Mellon University • June 19-22, 2017

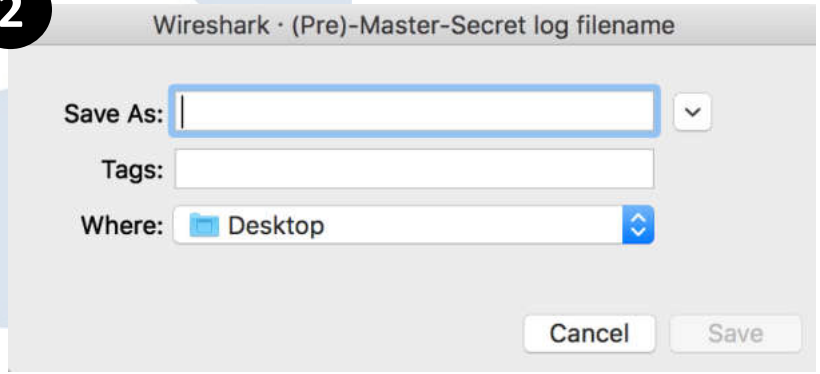
1



- Go back to SSL settings *(Edit/Preferences/Protocol/SSL/(Pre)-Master_Secret Log filename)*
- Click on Browse to select file

- Select TLS key *(your location of the flile)*
- Click Save
- Click OK

2



- Purpose: reads packet data from a given

- New command used:

Command	Purpose
-r	Provides pathname of capture file to be read in as a command line argument

Windows Command

C:[dir]>"C:\[dir]\Wireshark\tshark.exe" –r [drive]:\[dir]\southbound.pcap

Linux Command

tshark -r [Directory]/southbound.pcap

- Purpose: decrypt the first layer of encryption on the traffic by applying the SSL key found in the virtual switch memory

- New command used:

Command	Purpose
-o	Set a preference or recent value, overriding the default value and any value read from a preference/recent file

Windows Command

C:[dir]>"C:\[dir]\Wireshark\tshark.exe" –o ssl.keys_list:192.168.1.1,6633,openflow,rsa_p_key.pem –r [drive]:\[dir]\southbound.pcap

Linux Command

tshark -o "ssl.keys_list:192.168.1.1,6633,openflow,rsa_p_key.pem" -r SDN\ Files/southbound.pcap

- Purpose: decrypt the second layer of encryption on the traffic by applying the TLS key found in the virtual switch memory and the hello message of the network traffic

- New command used:

Command	Purpose
-o	Set a preference or recent value, overriding the default value and any value read from a preference/recent file

Windows Command

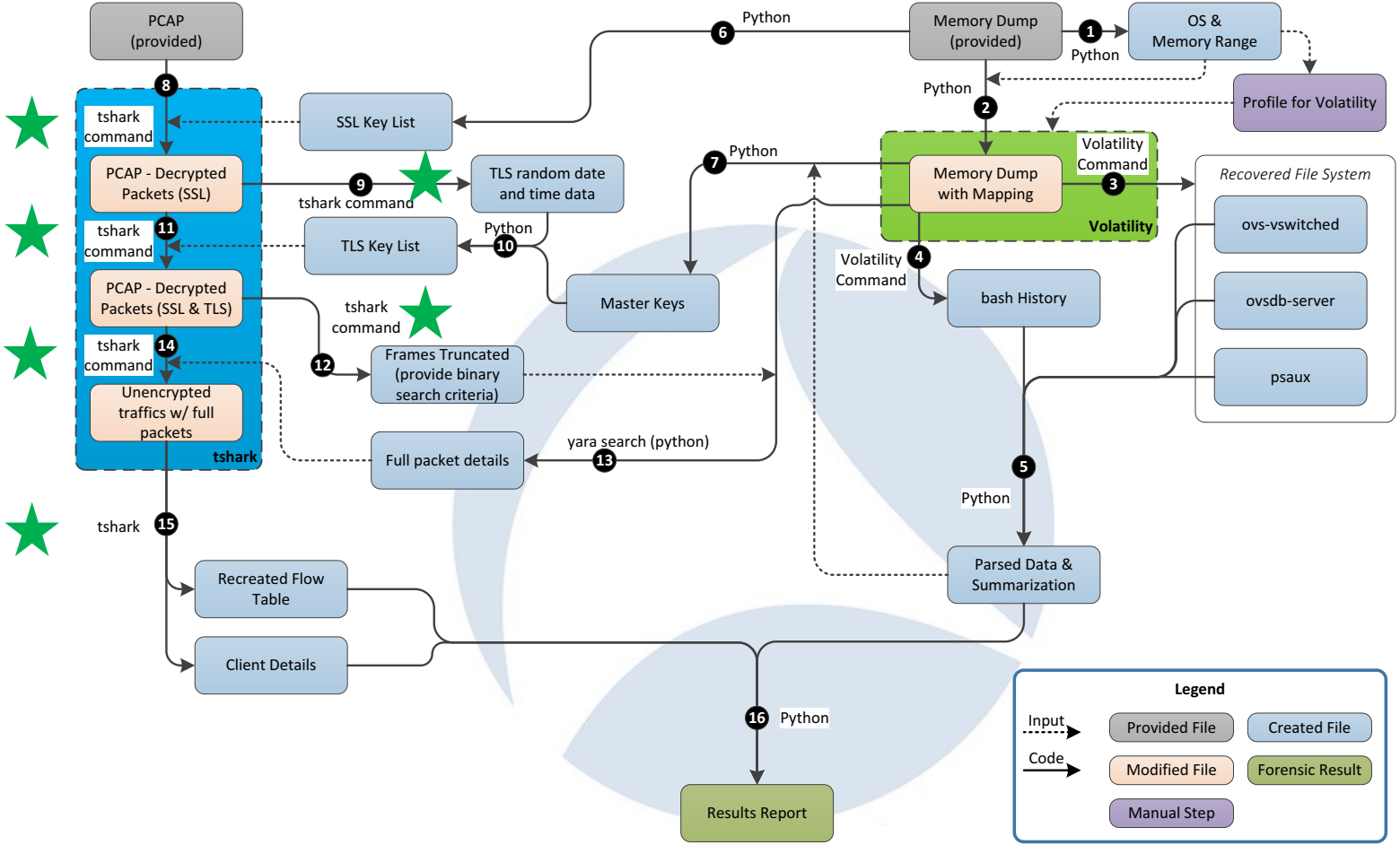
C:[dir]>"C:\[dir]\Wireshark\tshark.exe" –o ssl.keys_list:192.168.1.1,6633,openflow,rsa_p_key.pem -o ssl.keylog_file:keylog_file2.txt –r [drive]:\[dir]\southbound.pcap

Linux Command

tshark -o "ssl.keys_list:192.168.1.1,6633,openflow,rsa_p_key.pem"
-r SDN\ Files/southbound.pcap
-o ssl.keylog_file:keylog_file2.txt

tshark Wins!

Jupyter Demo



SharkFest'17 US • Carnegie Mellon University • June 19-22, 2017

Thanks & Question

Backup Slides