# SharkFest '16

## Forensic Network Analysis in the Time of APTs

June 16th 2016

Christian Landström

Senior IT Security Consultant | Airbus Defence and Space CyberSecurity

SharkFest '16 • Computer History Museum • June 13-16, 2016

# Topics

- Overview on security infrastructure
- Strategies for network defense and forensics
- A look at malicious traffic incl. Demos
- How Wireshark can help
- Best Practice Proactive / Reactive

# House Rules

# Tool-Box

**Defaults:**

Proxy servers with authentication

Logging, Monitoring, (SIEM)

Layers of Defense:

Firewalls / WAFs

Intrusion Detection / Intrusion Prevention

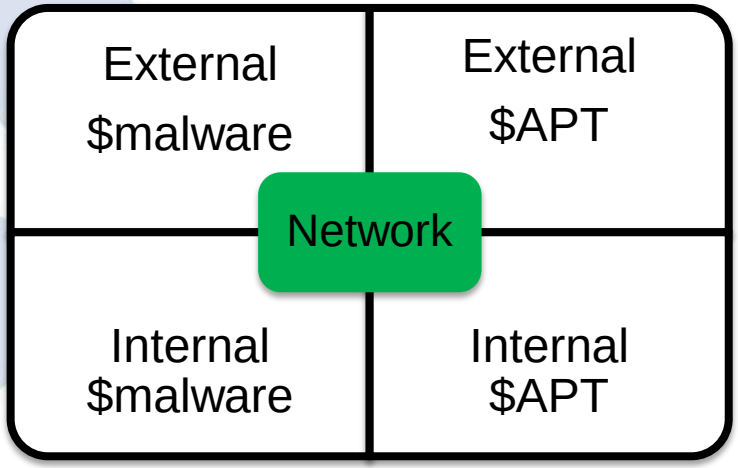
- NIDS/NIPS/HIDS/HIPS

Malware Sensors / Sandboxing / “APT-devices”

# Overview on sec. infrastructure

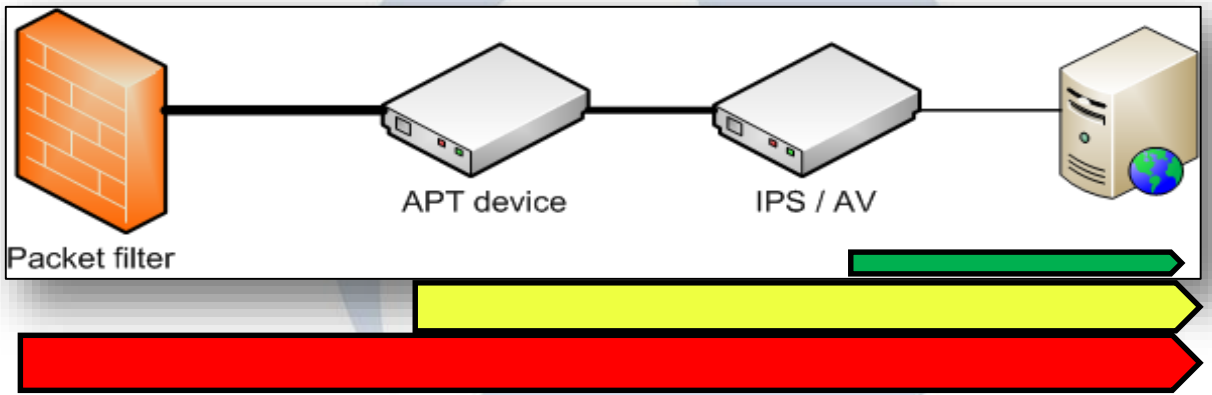
- Depending on
  → area of protection
  → type of attack - leaving out inside jobs (!!!)

Malicious Traffic types:
- External: Internet facing
- Internal: non-Inet facing



# Typical protection for DMZ systems:

Packet filter → IPS / APT device → local (host-)firewall



# What do companies expect

- Firewall protecting from all sorts of unwanted traffic towards internal systems
- IDS / IPS sending Alerts for all sorts of exploitation attempts and abnormal network traffic
- “APT” / Sandboxing devices to trigger on malicious code / malicious binary files
- Host IPS / Host Firewalls alerting any type of unwanted access, traffic or what not…

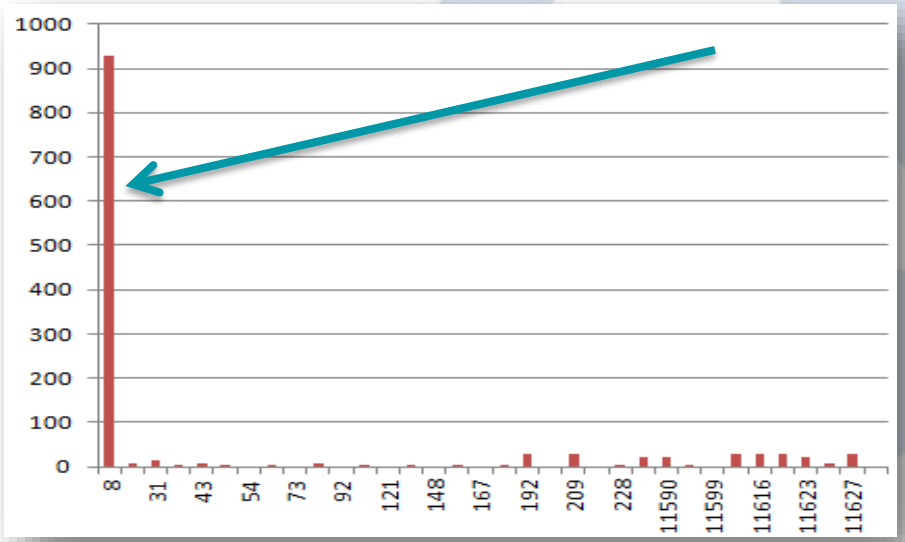
# Demo #1: DMZ Service

Monitoring the request size in this example reveals some huge request resulting in a new connection initiated by the FTP Server

| Source | Destination | Protocol | Size | Info |
|---|---|---|---|---|
| 192.168.163.130 | 192.168.163.128 | TCP | 74 | 41779→21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval= |
| 192.168.163.128 | 192.168.163.130 | FTP | 108 | Response: 220 3Com 3CDaemon FTP Server Version 2.0 |
| 192.168.163.130 | 192.168.163.128 | FTP | 1000 | Request: USER 5FFy8o^Geersu!2E,ND3?[4gz)M5V,cC_MzJuMv}a]1C<*[mF |
| 192.168.163.128 | 192.168.163.130 | TCP | 62 | 1086→4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 |
| 192.168.163.1 | 192.168.163.128 | TCP | 66 | 56571→21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 192.168.163.128 | 192.168.163.1 | FTP | 96 | Response: 220 3Com 3CDaemon FTP Server Version 2.0 |
| 192.168.163.1 | 192.168.163.128 | FTP | 70 | Request: USER anonymous |
| 192.168.163.128 | 192.168.163.1 | FTP | 87 | Response: 331 User name ok, need password |
| 192.168.163.1 | 192.168.163.128 | FTP | 75 | Request: PASS anon@anon.anon |
| 192.168.163.128 | 192.168.163.1 | FTP | 74 | Response: 230 User logged in |
| 192.168.163.1 | 192.168.163.128 | FTP | 60 | Request: SYST |
| 192.168.163.128 | 192.168.163.1 | FTP | 73 | Response: 215 UNIX Type: L8 |
| 192.168.163.1 | 192.168.163.128 | FTP | 60 | Request: FEAT |
| 192.168.163.128 | 192.168.163.1 | FTP | 76 | Response: 211- Feature listing |
| 192.168.163.128 | 192.168.163.1 | FTP | 88 | Response: MDTM |

# Demo #1: DMZ Service

Knowing your applications' behavior may lead to valid thresholds to reveal anomalies e.g. based on packet length, payload entropy or other factors



# External perimeter defense

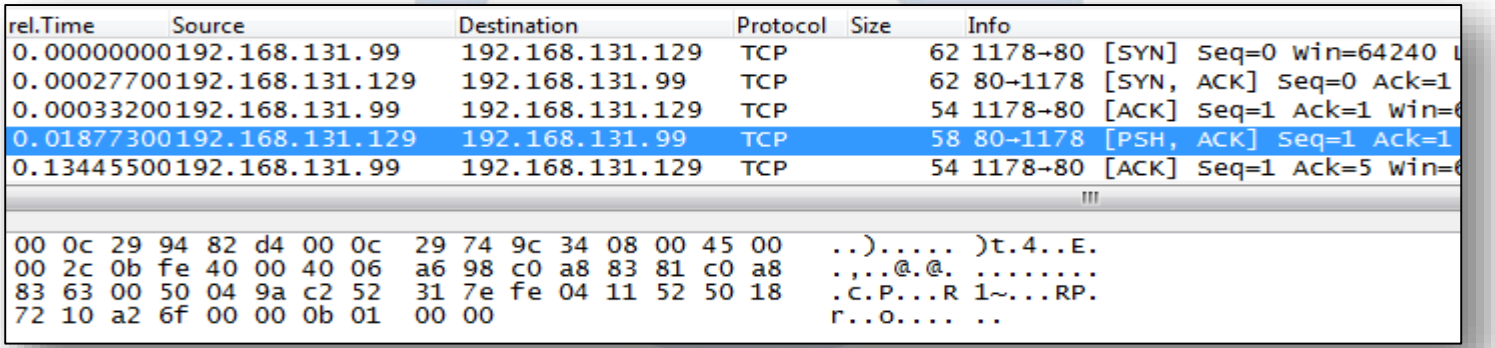
Perimeter defense: Monitoring all protocols

- Know your systems' configuration
- In-depth understanding of App behavior
- Monitor the events from sec. devices
- Correlate events after sec. alert

→ WebServer accessing other servers after "unsuccessful" exploit?

# Demo #2: “Encrypted” sessions

Watch for protocol anomalies e.g. missing HTTP dissector information on HTTP ports containing no valid requests or malformed data



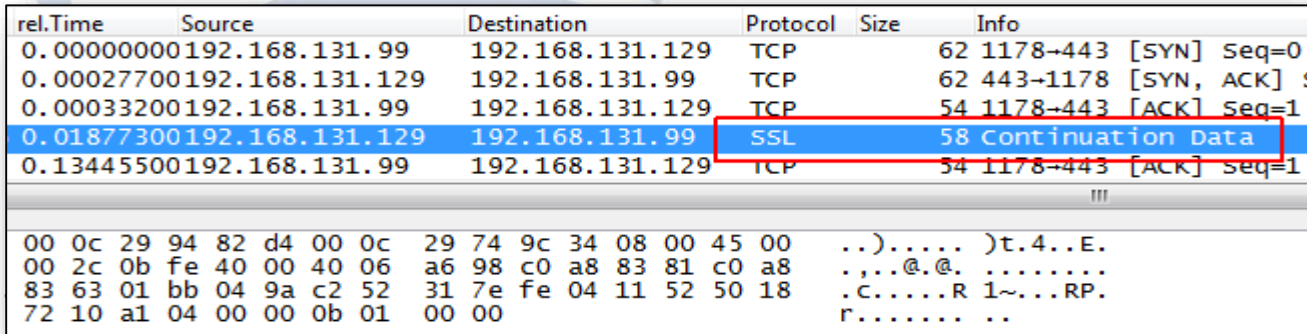
| rel.Time | Source | Destination | Protocol | Size | Info |
|---|---|---|---|---|---|
| 0.00000000 | 192.168.131.99 | 192.168.131.129 | TCP | 62 | 1178→80 [SYN] Seq=0 Win=64240 L |
| 0.00027700 | 192.168.131.129 | 192.168.131.99 | TCP | 62 | 80→1178 [SYN, ACK] Seq=0 Ack=1 |
| 0.00033200 | 192.168.131.99 | 192.168.131.129 | TCP | 54 | 1178→80 [ACK] Seq=1 Ack=1 Win=6 |
| 0.01877300 | 192.168.131.129 | 192.168.131.99 | TCP | 58 | 80→1178 [PSH, ACK] Seq=1 Ack=1 |
| 0.13445500 | 192.168.131.99 | 192.168.131.129 | TCP | 54 | 1178→80 [ACK] Seq=1 Ack=5 Win=6 |

```
00 0c 29 94 82 d4 00 0c  29 74 9c 34 08 00 45 00   ..)..... )t.4..E.
00 2c 0b fe 40 00 40 06  a6 98 c0 a8 83 81 c0 a8   .,..@.@. ........
83 63 00 50 04 9a c2 52  31 7e fe 04 11 52 50 18   .c.P...R 1~...RP.
72 10 a2 6f 00 00 0b 01  00 00                     r..o.... ..
```

# Demo #2: “Encrypted” sessions

Another example for pretended encrypted traffic not containing a valid SSL handshake

```
Sample: Using relative Sequence numbers try:
 tshark –r <tracefile> -Y "tcp.dstport==443 and
 tcp.len > 0 and tcp.seq == 1 and !ssl.record"
```



# The key question

Are you doing network forensics

a) To check whether there is something bad
b) To analyze something bad that is already known to be there

# Internal I

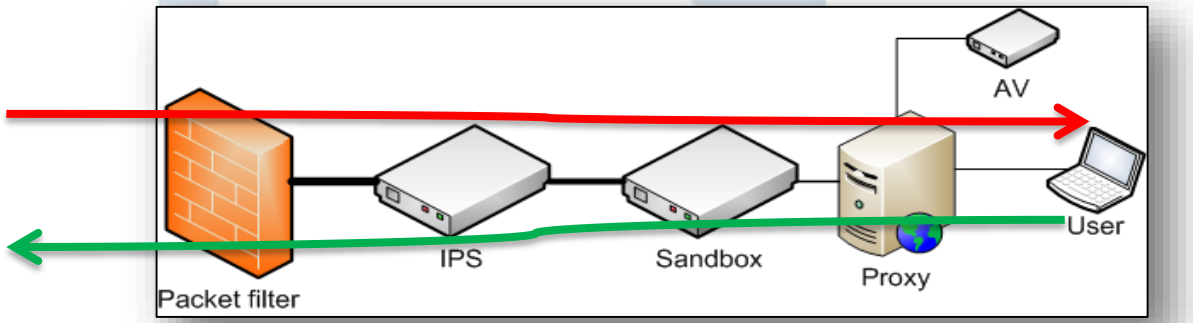
Incoming traffic critical and monitored

**But:**

Sessions going out are trusted Mail/Web/FTP etc.

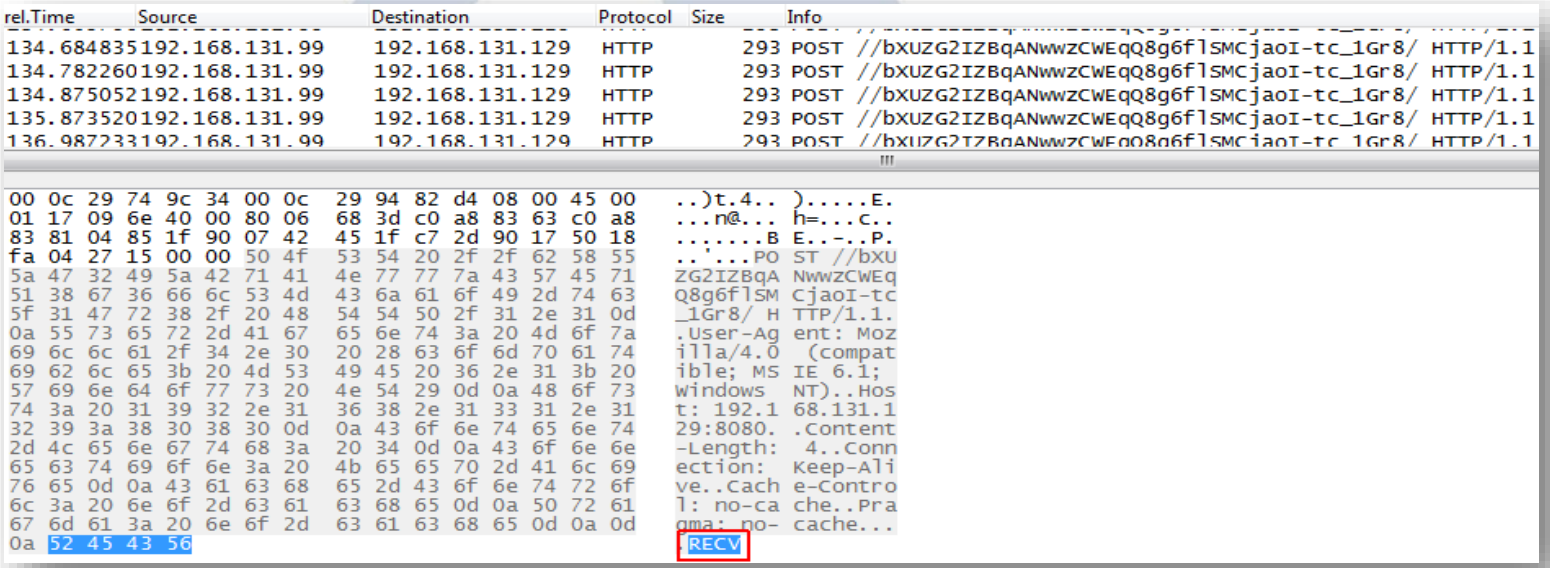
Internal traffic between “trusted devices”

How to spot outgoing malicious stuff?



# Demo #3: Surfing the web

Also valid protocol requests may hint for an anomaly based on irregular behavior or other indicators



| rel.Time | Source | Destination | Protocol | Size | Info |
|---|---|---|---|---|---|
| 134.684835 | 192.168.131.99 | 192.168.131.129 | HTTP | 293 | POST //bXUZG2IZBqANwwzCWEqQ8g6flSMCjaoI-tc_1Gr8/ HTTP/1.1 |
| 134.782260 | 192.168.131.99 | 192.168.131.129 | HTTP | 293 | POST //bXUZG2IZBqANwwzCWEqQ8g6flSMCjaoI-tc_1Gr8/ HTTP/1.1 |
| 134.875052 | 192.168.131.99 | 192.168.131.129 | HTTP | 293 | POST //bXUZG2IZBqANwwzCWEqQ8g6flSMCjaoI-tc_1Gr8/ HTTP/1.1 |
| 135.873520 | 192.168.131.99 | 192.168.131.129 | HTTP | 293 | POST //bXUZG2IZBqANwwzCWEqQ8g6flSMCjaoI-tc_1Gr8/ HTTP/1.1 |
| 136.987233 | 192.168.131.99 | 192.168.131.129 | HTTP | 293 | POST //bXUZG2IZBqANwwzCWEqQ8g6flSMCjaoI-tc_1Gr8/ HTTP/1.1 |

```
00 0c 29 74 9c 34 00 0c  29 94 82 d4 08 00 45 00   ..)t.4.. )....E.
01 17 09 6e 40 00 80 06  68 3d c0 a8 83 63 c0 a8   ...n@... h=...c..
83 81 04 85 1f 90 07 42  45 1f c7 2d 90 17 50 18   .......B E..-..P.
fa 04 27 15 00 00 50 4f  53 54 20 2f 2f 62 58 55   ..'...PO ST //bXU
5a 47 32 49 5a 42 71 41  4e 77 77 7a 43 57 45 71   ZG2IZBqA NwwzCWEq
51 38 67 36 66 6c 53 4d  43 6a 61 6f 49 2d 74 63   Q8g6flSM CjaoI-tc
5f 31 47 72 38 2f 20 48  54 54 50 2f 31 2e 31 0d   _1Gr8/ H TTP/1.1.
0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   .User-Ag ent: Moz
69 6c 6c 61 2f 34 2e 30  20 28 63 6f 6d 70 61 74   illa/4.0  (compat
69 62 6c 65 3b 20 4d 53  49 45 20 36 2e 31 3b 20   ible; MS IE 6.1;
57 69 6e 64 6f 77 73 20  4e 54 29 0d 0a 48 6f 73   Windows  NT)..Hos
74 3a 20 31 39 32 2e 31  36 38 2e 31 33 31 2e 31   t: 192.1 68.131.1
32 39 3a 38 30 38 30 0d  0a 43 6f 6e 74 65 6e 74   29:8080. .Content
2d 4c 65 6e 67 74 68 3a  20 34 0d 0a 43 6f 6e 6e   -Length:  4..Conn
65 63 74 69 6f 6e 3a 20  4b 65 65 70 2d 41 6c 69   ection:  Keep-Ali
76 65 0d 0a 43 61 63 68  65 2d 43 6f 6e 74 72 6f   ve..Cach e-Contro
6c 3a 20 6e 6f 2d 63 61  63 68 65 0d 0a 50 72 61   l: no-ca che..Pra
67 6d 61 3a 20 6e 6f 2d  63 61 63 68 65 0d 0a 0d   gma: no- cache...
0a 52 45 43 56                                     .RECV
```

# Internal II

Big issue: Lateral movement and other post-infection activities

- Internal scanning / enumeration
- Access to internal applications
- brute force attempts
- legitimate access with stolen credentials

→ Mostly depending on log files from internal sources

# Baselining / Anomaly detection

Knowing your application behavior / network flows is critical to spotting malicious events

- Might be easy for default applications

  → Statistics: Conversation e.g.

- How about special applications?

# Demo #4: Baselining sample

- Especially difficult if application payload types unknown or difficult to baseline

```
# tshark -r Trace1.pcap -Y udp -Tfields -e data | more
4b417947534b675341414274615735706247467459 6d3841524739
e1650518e41793d5abb03d
755d021f5cf975c6342cc14f84caf5e0b863
e1680231b0aee0ecbb648c0a4b14167412cbfb16356e8b6b76db
755f02cf93f622f368d2fef70bf71c5e5f85a8e297eb79795ac04f
```

Legitimate example Skype

```
# tshark -r Trace2.pcap -Y udp -Tfields -e data | more
10a6b286d9736aae21afc2ddf005f6125f66633de613a63e46
10a6b286d9736aae21afc2ddf005f6125f66633de613a63e46
10a7
10a0b286d9736aae21afc2ddf005f6125f66633de613a63e46
10b15a78
10bf281d1581812c38ee0e0d90c18f2e5458bbc25bc030b0
10a1530e1598ba7ad499afea4ca126827f07de483537d0ad14c0be
```

Malicious example Peacomm.C malware

# Baselining approaches e.g. Web

- Many approaches for finding unknown sources of malicious activity
- Sample: domain lists -> diff approach
  - Cat I : Clean or already infected
  - Cat II : newly infected
- Timely Diff's -> approach new infections / applications

# How Wireshark can help

- Better understanding of your application behavior
- Scripted generation of baselining data
- Long-term comparison of network traces for detecting abnormal changes
- Incident Analysis Results can lead to good rules for IDS/IPS and other appliances

**!! NO excuse for not having good log files !!**

# Where's the catch?

- Depending on the type of intrusion you're facing, different approaches are needed
- Criticality differs:
  - Standard Malware
  - Advanced Malware
  - Targeted dedicated Malware with strong external c2c and typical behaviour
  - Advanced compromise relying on classic malware
  - Advanced compromise using targeted tooling and completely unique software and leveraging max. legit looks

# Demo #5: How Wireshark can help

DNS answers for localhost IP can lead to inactive c2c system

**Beware**: Also used for lots of valid reasons e.g. SPAM checking

```
tshark -r 127.0.0.x.pcap -Tfields -e dns.qry.name | grep -v -E
  "(<valid1>|<valid2>)" | sort | uniq -c | more
[…]
      1 xxxxxxx.mcafee.com
      1 yyyyyyy.mcafee.com
    147 <malicious1>.is-cert.com
    148 <malicious2>.dnsas.com
    146 <malicious3>.ddns-ip.com
    148 <malicious4>.ddns-office.com
    148 <malicious5>.ddns.com
```

# Demo #5: How Wireshark can help

Alternative: tshark –r 127.0.0.x.pcap –q –z hosts

Difference: Multiple answers containing same IP address in dns.a NOT listed

```
tshark -r 127.0.0.x.pcap –q –z hosts
[…]
127.0.0.1       {…}.ddns-ip.com
127.0.0.100     {…}.xxxxxx.mailshell.net
127.0.0.255     {…}
127.0.0.128     {…}
```

# Recommendation: Malware Traffic Analysis

http://malware-traffic-analysis.net/index.html

Brad Duncan

# Recommendation: Network Forensics Workshop

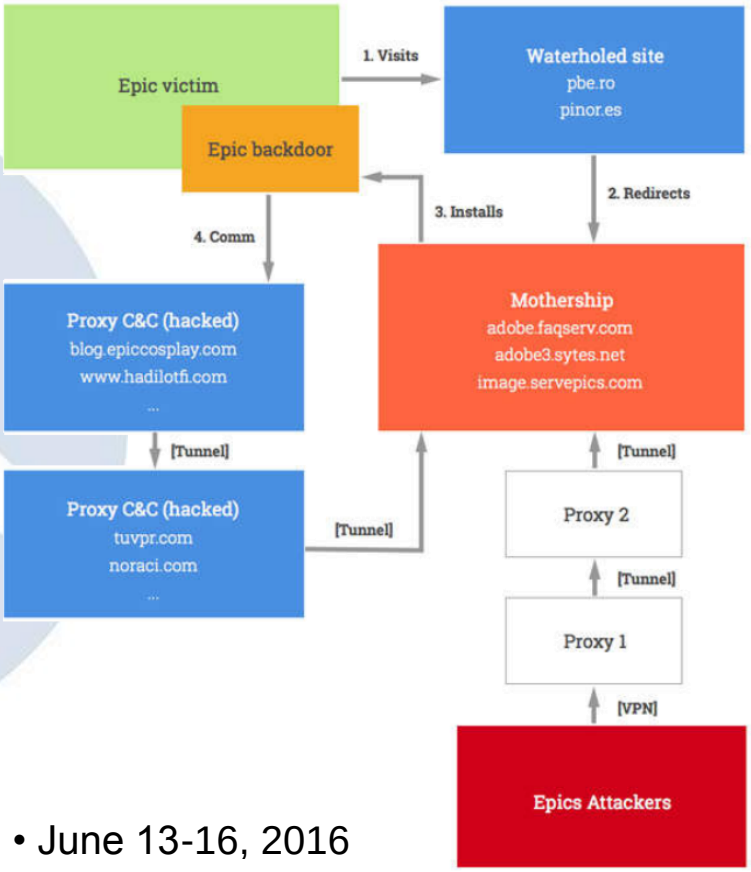
https://www.first.org/_assets/conf2015/networkforensics_virtualbox.zip

PDF: first_2015_-_hjelmvik-_erik_-_hands-on_network_forensics_20150604

# Tracing back

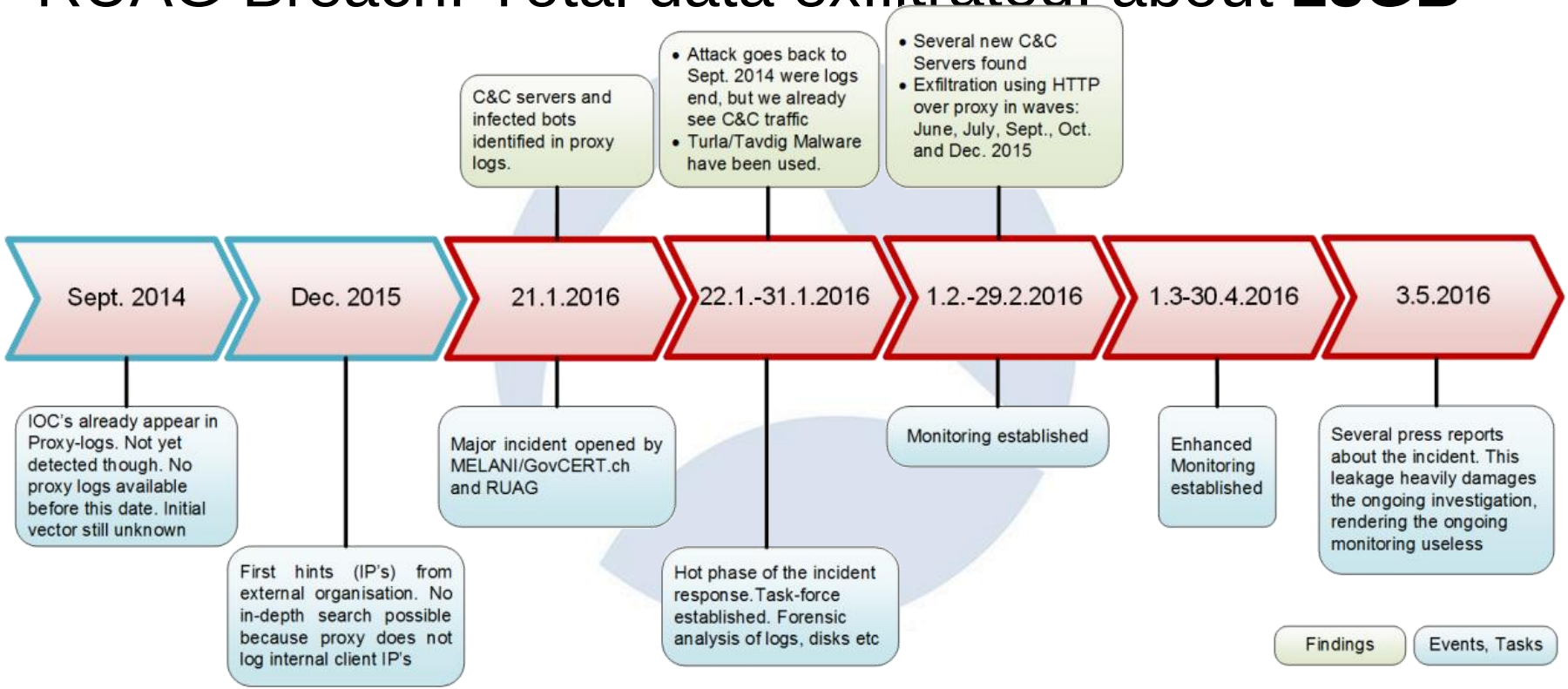
- Difficult at best when serious

- Image from Kaspersky Report about Epic Turla



# The “Time” Factor

## RUAG Breach: Total data exfiltrated: about **23GB**



# Bringing it to the limit

Maximizing legitimate traffic types and applications - e.g. Hammertoss

Check FireEye Report on APT29 -> search engine

# Monitoring Networks - Proactive

- Use NetFlow/OpenFlow to monitor meta data
  - Set up alerts for unusual patterns
- Use IDS/IPS with optimized signatures
  - Reduce false positives as much as possible
- Set up Passive DNS / Passive SSL recording servers
  - Helps in tracking down name resolution and certificate history

# Monitoring Networks - Reactive

- Forensic analysis on full packet captures
  - Has to be recorded before something happened, of course
  - Carefully selected locations, e.g. Internet outbreaks

- Use NetFlow/OpenFlow for meta data
  - Long term storage for forensic searches, e.g. „where did the attacker connect to from the infected system?“

- Use IDS/IPS as custom IoC alarm system
  - Write custom IDS rules for known **I**ndicators **o**f **C**ompromise from Wireshark Analysis results

# Detecting malicious traffic

- Forget „silver bullets“
  → there is no “*showmethebadstuff*” filter

- Attackers may hide in plain sight (DNS, HTTP(S), FTP,...)
- Filter out positives
  - E.g. Alexa 1 Million
  - Known update sites: OS, AV, Vendors

# Final Words

- Network defense is a 24/7 challenge
- Attackers only need to succeed once, defenders would need 100% success

  - Read as: it‘s not „if“ but „when“ an attack will succeed.
  - **Expect successful attacks on your network.**

- Keep searching
  - It‘s a continuous task
  - Don‘t just wait for some alarm to go off

# !! Thank you for attending !!

Questions?

---

eMail: landi@packet-foo.com
Web: www.packet-foo.com
Twitter: @0x6C616E6469