Suricata Tutorial

FloCon 2016

OPEN INFORMATION SECURITY FOUNDATION

Agenda

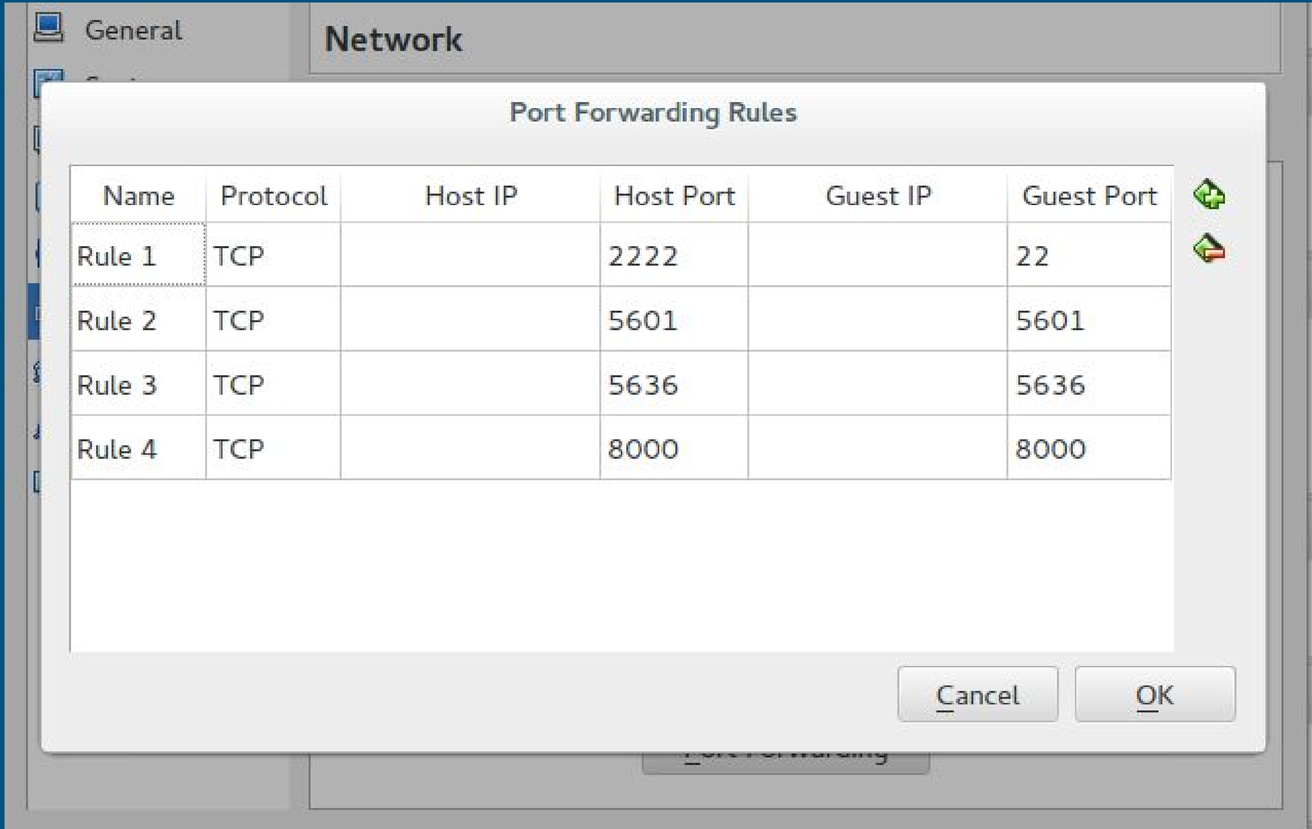
- Setup
- Introduction to Suricata
- Suricata as a SSL monitor
- Suricata as a passive DNS probe
- Suricata as a flow probe
- Suricata as a malware detector

VirtualBox setup

- File -> Preferences
  - Apple: ‘VirtualBox -> Preferences’
- Network -> Host Only Network (tab)
- Add network vboxnet0

VirtualBox Port Forwards

- 2222 SSH
- 5601 Kibana4
- 5636 Evebox
- 8000 Scirius



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP		2222		22
Rule 2	TCP		5601		5601
Rule 3	TCP		5636		5636
Rule 4	TCP		8000		8000

Setup

- We have USB keys with OVA files
- Please copy to local disk first
- Pass on USB key
- File -> Import Appliance. Select the OVA file.
- Username “suricata”. Password “suricata”
- ssh suricata@localhost -p2222

About us

- Eric Leblond - Freedom Fries
- Victor Julien - Cheese and Tulips

About us

- Victor Julien
    - Suricata lead developer
    - Open Source Hippie
- Eric Leblond
    - Suricata core developer
        - packet acquisition
        - unix socket
        - redis
    - Stamus Networks co-founder
    - Netfilter coreteam member

about OISF

- Mission
- Funding
- Support
- Code
- Community

Our Mission

The **Open Information Security Foundation** is a US based 501(c)3 **non-profit foundation** organized to build **community** and to support **open-source** security technologies like Suricata, the world-class IDS/IPS engine.

OISF
OPEN INFORMATION SECURITY FOUNDATION

OISF’s Funding

- Consortium Members - Platinum, Gold, Bronze… new “Start-Up” level coming.
- Grant with Department of Energy
- Suricata Training Events

Suricata Community Events

- 2-Day Trainings - West Coast (US), East Coast (US), Europe
- Developer Training - September 12th, Paris
- Suricata User Conference - November 9-11, Washingon, DC

www.oisf.net for information!

Note about the PCAPS

- taken with permission from malware-traffic-analysis.net
- many thanks to Brad at malware-traffic-analysis.net

Introduction to Suricata

OISF

OPEN INFORMATION SECURITY FOUNDATION

Who still knows their network?

- Increasing complexity
- BYOD
- IoT
- VM's and containers
- ICS/SCADA

Suricata is an engine for...

Network Intrusion Detection

Network Intrusion Prevention

Network Security Monitoring

IDS

- Intrusion Detection System
- Passive
- Out of line
- On tap or span port

IPS

- Intrusion Prevention System
- Active
- Inline
- Router or bridge

NSM

- Network Security Monitoring
- Not ‘just’ generating alerts, but also informational events like HTTP requests, TLS transfers, etc
- Full Packet Capture (FPC) for being able to dig deep into traffic if necessary
- Produces LOTS of data

Suricata Ecosystem

- Distributions
  - SELKS & Amsterdam
  - SecurityOnion
  - pfSense & OPNsense
- Management tools
  - Evebox
  - Scirius
  - Kibana
- Event processing
  - Mobster
  - Barnyard2
  - Logstash

Suricata’s main features

- Inspect traffic for known bad using extended Snort language
- Lua based scripting for detection
- Unified JSON output for easy post-processing
- File extraction
- Scalable through multi-threading

Technical Features

- IPv4/IPv6, defrag, flow tracking
- TCP tracking, reassembly
- Port independent protocol detection
- Stateful HTTP, SMTP, DNS, TLS parsing
- File extraction for HTTP, SMTP
- Rule language additions: SSH, TLS, file names, type & md5
- IP Reputation, GeoIP, IP list support
- Lua scripting for extending detection and outputs
- (Net)flow like output logging

Suricata and performance

- Scalability via multithreading
    - Almost linear scalability
    - Around 450-650 Mbps per core
- 1Gbps
    - Multicore required
    - Straight setup
- 10Gbps
    - Possible on commodity hardware
    - Serious tuning needed

Suricata 2.0

- Current Stable
- Eve, an all JSON alert and event stream
- For use with Splunk,Logstash and native JSON log parsers
- DNS parser, matcher and logger
- “NSM runmode” -> only events, no rules and alerts

Suricata 3.0

- In Release Candidate cycle. Due January 27th.
- SMTP file extraction and logging
- Performance & scalability!
- Lua scripting++
- Multitenancy
- Redis output
- Flow logging

Rulesets

- 2 main sources of IDS rules
  - Emerging Threats (Proofpoint)
  - VRT/Talos (Sourcefire/Cisco)
- Both have free and paid sets
- Emerging Threats is optimized for Suricata

Introduction to SELKS

- Ready to use Linux distribution featuring
  - Suricata 3.0*
  - Elasticsearch: database
  - Logstash: data pipeline
  - Kibana: dashboard and visualization interface
  - Scirius: suricata ruleset management
- Availability
  - As a Live and Installable ISO
  - GPLv3

Introduction to “Amsterdam”

- Goals
  - Provide features of SELKS via docker containers
  - Objective is super fast installation
- Amsterdam provides
  - Latest ELK and suricata
- Basic setup sniffing traffic on physical host:
  - pip install amsterdam
  - amsterdam -d flocon -i wlan0 setup
  - amsterdam -d flocon start
  - firefox http://localhost:8000

OISF
OPEN INFORMATION SECURITY FOUNDATION

Starting “Amsterdam”

- boot VM
- login directly or “ssh suricata@localhost -p2222”
- run “amsterdam -d flocon start”
- open a new SSH connection to the VM
- in ~/flocon the various “Amsterdam” components have their output dirs

Testing Amsterdam

- “Amsterdam” runs on the “eth0” in the VM, connected to the host only network
- from the VM we can “replay” pcaps to “Amsterdam”
- sudo tcpreplay -i eth0 pcaps/2015-01-09-traffic-analysis-exercise.pcap
- now tail -f ~/flocon/suricata/stats.log

Suricata commandline

- General Suricata commands
  - -v, -h
  - --build-info
  - -i eth0
  - - r <pcap file>
  - -S <rule file>
  - -T -> test config & rules
- To run command inside running container:
  - `docker exec flocon_suricata_1  suricata -V`

Suricata as a TLS monitor

TLS tracking in Suricata

- Suricata tracks SSL/TLS sessions
- No decryption capabilities
- Looking at TLS still valuable
  - heartbleed
  - certificate validation

TLS Logging

- subject
- issuer
- fingerprint
- server name indication (SNI)
- protocol version

SSL Logging Example

```
{"timestamp":"2016-01-06T11:20:31.431359+0100","flow_id":105716325071680,"
in_iface":"eth0","event_type":"tls","src_ip":"192.168.1.6","src_port":48952,"
dest_ip":"173.194.65.132","dest_port":443,"proto":"TCP","tls":{"subject":"C=US,
ST=California, L=Mountain View, O=Google Inc, CN=*.googleusercontent.com",
"issuerdn":"C=US, O=Google Inc, CN=Google Internet Authority G2", "fingerprint":"
b2:e7:5a:d1:e4:3a:a9:a8:37:f5:13:b0:1a:88:70:a2:60:fe:8a:4a", "sni":"lh3.
googleusercontent.com","version":"TLS 1.2"}}
```

Replay pcap containing TLS

- Download the pcap as suricata user
  - wget http://home.regit.org/~regit/flocon-tls.pcap
- Replay the pcap
  - sudo tcpreplay -i eth0 flocon-tls.pcap
  - Wait 90s for completion

Usage in Kibana

- Create the following visualization and add them to a dashboard
  - Pie with TLS version
  - Bar diagram with Top issuer DNs splitted by server IP
- Demonstration
  - Top SNI timeline with point being unique servers

Using jq

- JQ is a command line tool to operate filtering and transformation on JSON
- Install it
    - sudo apt-get install jq
- Basic usage is to enhance format
    - cd flocon/suricata
    - cat eve.json | jq ‘.’
    - cat eve.json | jq -c ‘.’
    - tail -f eve.json | jq -c ‘.’

Using jq

Select only TLS events

```
cat eve.json | jq 'select(.event_type=="tls")'
```

Use jq to show only sni and issuerdn

```
cat flocon/suricata/eve.json | jq '{ sni:.tls.sni, issuerdn:.tls.issuerdn}'
```

Find self signed certificates

```
cat eve.json | jq 'select(.event_type=="tls" and .tls.subject==.tls.issuerdn)'
```

Using TLS detection

- keywords to match on issuerdn, subject, fingerprint
- combine with protocol detection for TLS on non-std ports
- HTTP & other protocols on port 443
- Lua

Alert example:

```
alert tls any any -> $SERVERS any ( tls.issuerdn:!"C=NL, O=Staat der Nederlanden, CN=Staat der Nederlanden Root CA";)
```

Alerting on self-signed certificates

The rule:

alert tls any any -> any any (msg:"SURICATA TLS Self Signed Certificate"; flow:established; luajit:self-signed-cert.lua; tls.store; sid:999666111; rev:1;)

The script

```
function match(args)
    version, subject, issuer, fingerprint = TlsGetCertInfo();
    if subject == issuer then
        return 1
    else
        return 0
    end
end
```

Exercise: tls lua script (1/2)

- Download the ruleset on laptop
  - http://home.regit.org/~regit/tls-self-signed.tgz
- Connect to
  - http://localhost:8000
- Click on “Sources”, then “add source”
- Select Archive + Upload
- Click “Suricata,” then “ruleset actions”
- Select “build” and ”push”

Exercise: tls lua script (2/2)

- Activate tls-store in yaml:
  - sudo vi flocon/config/suricata/suricata.yaml
  - Switch enabled to yes for tls-store
- Restart suricata
  - docker restart flocon_suricata_1
- Replay flocon-tls.pcap
- Refresh suricata page of scirius to see alerts
- Check that certificate are created
  - openssl x509 -in flocon/suricata/1452462998.778376-1.pem -text

Suricata as a passive DNS probe

Suricata DNS tracking

- Suricata does stateful DNS tracking for UDP and TCP
- Stateful in the sense that requests and responses are matched

Suricata DNS Logging

- log DNS transactions in EVE
  - file
  - syslog
  - redis
  - unix socket
  - lua script(s)
- log the data of various record types
  - A, AAAA
  - MX, PTR
  - TXT

Exercise: NXDOMAIN

- Lets try to look into NXDOMAIN responses
- tcpreplay -M1 -i eth0 pcaps/2015-02-15-traffic-analysis-exercise.pcap
- Kibana:
    - In Discover tab, search “event_type:dns”, then save the search as “DNS events”
    - In Visualize tab, select Pie Chart. From Saved Search. Select “DNS events”
    - In Buckets (left) select split slices, Aggregation “terms”, select field “dns.rcode.raw”
    - Save as “DNS Error”
    - In Dashboard tab: “Add Visualization” and select “DNS Error”
    - In Dashboard tab: “Add Visualization”, “Searches” tab, then “DNS Events”

Exercise: DNS types pie graph

- Create a pie diagram of the top 10 used DNS types
- Hint: use dns.rrtype.raw

Exercise: show DNS names with TTL < 100

- Create visualization in Kibana
- Hint: search for “dns.ttl:[0 TO 99]”

Suricata as a flow probe

Suricata flow tracking

- Suricata keeps ‘flow’ records
  - bidirectional
  - uses 5 or 7 tuple depending on VLAN support
  - used for storing various ‘states’
    - TCP tracking and reassembly
    - HTTP parsing
- Flow records are updated per packet
- Flow records time out

Suricata Flow Output

- Two different outputs with similar data
- ‘flow’
  - Bidirectional
- ‘netflow’
  - Unidirectional
- Data contained
  - IP tuple
  - Duration and volumetry
  - Application layer info

Suricata Flow Logging

- Flow Hash management is done asynchronously
- A flow is timed out after no packets have been seen for it for some time
- When a flow is timed out, it can be logged
- The logging API allows for logging to:
  - file
  - syslog
  - redis
  - unix socket
  - lua script(s)
  - or any combination of the above

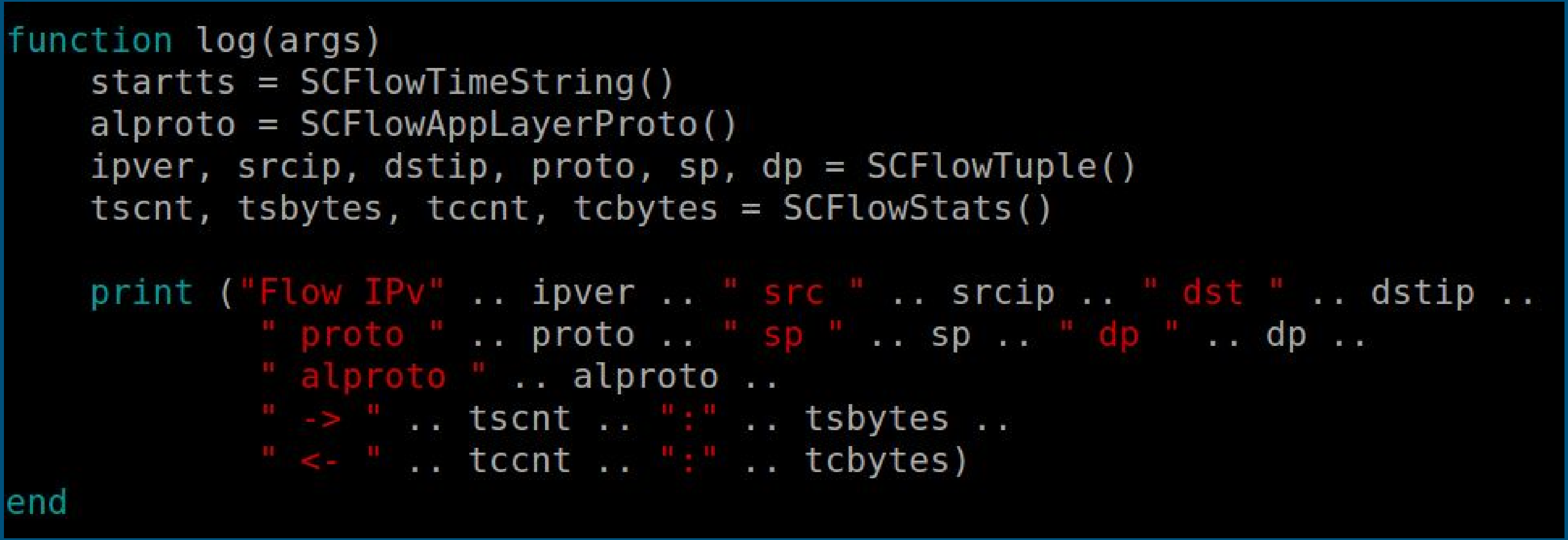
Flow output records

- bidirectional
- IP protocol, source, destination, source port, destination port
- packet count, bytes count
- start time stamp (first packet), end time stamp (last packet)
- L7 protocol as detected based on traffic content
- TCP
  - flags seen
  - state at flow end

Flow Logging Example

```
{"timestamp":"2009-11-11T02:01:04.731888+0100","flow_id":105716325086112,"
event_type":"flow","src_ip":"192.168.2.9","src_port":2432,"dest_ip":"
174.133.12.162","dest_port":80,"proto":"TCP","app_proto":"http","flow":
{"pkts_toserver":26,"pkts_toclient":36,"bytes_toserver":1885,"bytes_toclient":
47934,"start":"2009-11-11T02:01:02.937818+0100","end":"2009-11-11T02:01:
04.731888+0100","age":2,"state":"closed","reason":"shutdown"},"tcp":{"tcp_flags":"
1b","tcp_flags_ts":"1b","tcp_flags_tc":"1b","syn":true,"fin":true,"psh":true,"ack":true,"
state":"closed"}}
```

Using Lua scripts for output



```
function log(args)
    startts = SCFlowTimeString()
    alproto = SCFlowAppLayerProto()
    ipver, srcip, dstip, proto, sp, dp = SCFlowTuple()
    tscnt, tsbytes, tccnt, tcbytes = SCFlowStats()

    print ("Flow IPv" .. ipver .. " src " .. srcip .. " dst " .. dstip ..
            " proto " .. proto .. " sp " .. sp .. " dp " .. dp ..
            " alproto " .. alproto ..
            " -> " .. tscnt .. ":" .. tsbytes ..
            " <- " .. tccnt .. ":" .. tcbytes)
end
```

Inject traffic in the VM

- sudo tcpreplay -M1 -i eth0 pcaps/2015-*
    - starts a slow replay
- tail -f ~/flocon/suricata/eve.json | jq -c ‘select(.event_type==”flow”)’

Kibana visualization

- Timeline with flow count
- Timeline with mean value of flow duration
- Timeline with mean value of flow duration per protocol
- Donut with source, proto, destination

Scripting flow events in Python

- JSON module is official
- Deserialization via a single function
- Access to JSON like you access to a dictionary

Scripting JSON: example in Python

```
import json

with open('eve.json') as f:

    for line in f:

        event = json.loads(line)

        print event[‘event_type’]
```

Python scripting

Display events in the classical format

src ip:src port -> dst ip:dst port

Scripting JSON: example in Python

```
import json

with open('eve.json') as f:

    for line in f:

        event = json.loads(line)

        if event['event_type'] == 'flow':

            print("%s:%d --> %s:%d" % (event['src_ip'], event['src_port'], event['dest_ip'], event['dest_port']))
```

Python scripting

Display events in the format

```
src ip:src port -> dst ip:dst port [pkt_count]
```

Scripting JSON: example in Python

```
import json

with open('eve.json') as f:

    for line in f:

        event = json.loads(line)

        if event['event_type'] == 'flow':

            print("%s:%d --> %s:%d [pkts %d]" % (event['src_ip'], event['src_port'], event['dest_ip'], event['dest_port'], event['flow']['pkts_toserver']))
```

Python scripting

Add application protocol or layer 3 protocol if not available to the display

Scripting JSON: example in Python

```
with open('/tmp/eve.json') as f:
    for line in f:
        event = json.loads(line)
        if event['event_type'] == 'flow':
            if event.has_key('app_proto'):
                app_proto = event['app_proto']
            else:
                app_proto = event['proto']
            print("%s:%d - %s -> %s:%d [pkts %d]" % (event['src_ip'], event['src_port'], app_proto, event['dest_ip'], event['dest_port'], event['flow']['pkts_toserver']))
```

Suricata as a malware detector

OPEN INFORMATION SECURITY FOUNDATION

Suricata as a malware detector

- Rule/signature based detection
- More the ‘traditional’ IDS functionality
- Emerging Threats ruleset has strong focus malware
  - landing pages
  - CnC
  - Lua detect scripts for infections
    - https://github.com/EmergingThreats/et-luajit-scripts
  - “Open” version loaded by default in “Amsterdam”

Start your replay engines

- sudo tcpreplay -M1 -i eth0 pcaps/2015-*
    - starts a slow replay
- tail -f ~/flocon/suricata/fast.log
- tail -f ~/flocon/suricata/eve.json | jq -c ‘select(.event_type==”alert”)|.alert’

Bonus

```
cat ~/flocon/suricata/eve.json | jq -c 'select(.alert.signature=="ET POLICY Outdated Windows Flash Version IE")|.payload' -r|base64 -d|grep -i flash
```

It’s a bit dangerous, so be careful

Short Demo of Evebox

- Evebox is a front-end to ElasticSearch with EVE data
- To try it, add a port-forwarding rule to VirtualBox for TCP/5636
- I’ll give a quick demo
- Try yourself at http://localhost:5636

Exercise: show Alerts on map

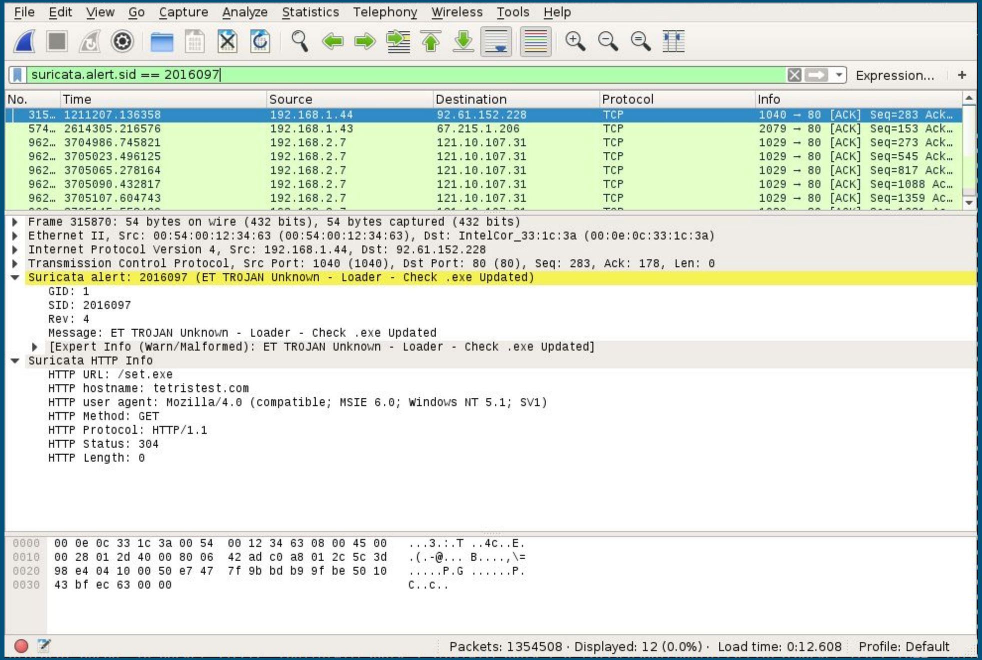
- In visualization, use Tile Map
- Use “Geo Coordinates”

Unix socket runmode

- A way to analyse fast a huge amount of pcap files
  - Coming from a honeypot
  - …
- Limitation in pcap reading mode
  - Detection engine optimisation can take 30 s or more
  - We need to skip this part
- In unix socket mode, suricata
  - Open a unix socket
  - wait for pcap file to analyse
  - output is done in specified directory

Showing Alerts in Wireshark

- Add EVE info to wireshark
- Done via suriwire plugin
- https://github.com/regit/suriwire


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

```
suricata.alert.sid == 2016097
```

```
No.    Time               Source         Destination     Protocol  Info
315…  1211207.136358     192.168.1.44   92.61.152.228   TCP       1040 → 80 [ACK] Seq=283 Ack…
574…  2614305.216576     192.168.1.43   67.215.1.206    TCP       2079 → 80 [ACK] Seq=153 Ack…
962…  3704986.745821     192.168.2.7    121.10.107.31   TCP       1029 → 80 [ACK] Seq=273 Ack…
962…  3705023.496125     192.168.2.7    121.10.107.31   TCP       1029 → 80 [ACK] Seq=545 Ack…
962…  3705065.278164     192.168.2.7    121.10.107.31   TCP       1029 → 80 [ACK] Seq=817 Ack…
962…  3705090.432817     192.168.2.7    121.10.107.31   TCP       1029 → 80 [ACK] Seq=1088 Ac…
962…  3705107.604743     192.168.2.7    121.10.107.31   TCP       1029 → 80 [ACK] Seq=1359 Ac…
```

```
▸ Frame 315870: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
▸ Ethernet II, Src: 00:54:00:12:34:63 (00:54:00:12:34:63), Dst: IntelCor_33:1c:3a (00:0e:0c:33:1c:3a)
▸ Internet Protocol Version 4, Src: 192.168.1.44, Dst: 92.61.152.228
▸ Transmission Control Protocol, Src Port: 1040 (1040), Dst Port: 80 (80), Seq: 283, Ack: 178, Len: 0
▾ Suricata alert: 2016097 (ET TROJAN Unknown - Loader - Check .exe Updated)
    GID: 1
    SID: 2016097
    Rev: 4
    Message: ET TROJAN Unknown - Loader - Check .exe Updated
  ▸ [Expert Info (Warn/Malformed): ET TROJAN Unknown - Loader - Check .exe Updated]
▾ Suricata HTTP Info
    HTTP URL: /set.exe
    HTTP hostname: tetristest.com
    HTTP user agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
    HTTP Method: GET
    HTTP Protocol: HTTP/1.1
    HTTP Status: 304
    HTTP Length: 0
```

```
0000  00 0e 0c 33 1c 3a 00 54  00 12 34 63 08 00 45 00   ...3.:.T ..4c..E.
0010  00 28 01 2d 40 00 80 06  42 ad c0 a8 01 2c 5c 3d   .(.-@... B....,\=
0020  98 e4 04 10 00 50 e7 47  7f 9b bd b9 9f be 50 10   .....P.G ......P.
0030  43 bf ec 63 00 00                                  C..c..
```

Packets: 1354508 · Displayed: 12 (0.0%) · Load time: 0:12.608   Profile: Default

PCAP credit: malware-traffic-analysis.net

Supporting Suricata

- Contribute to Suricata
- Become an OISF Consortium Member
- Host one of our 2-day Suricata Training Events
- Put us in touch with Trainers and (always!) Developers
- Follow Us - @OISFoundation and @Suricata_IDS
- Sponsor the 2016 Suricata User Conference - Washington, DC

5 Day Developer Training

- Paris, France
- Hosted by Mozilla
- Week of September 12th

mozilla

JOIN US!

2nd Annual Suricata User Conference

November 9 - 11, 2016

www.oisfevents.net

Thank You!

The Open Information Security Foundation

www.oisf.net

Suricata

www.suricata-ids.org

OPEN INFORMATION SECURITY FOUNDATION