Suricata 2.0, Netfilter and the PRC

Éric Leblond

Stamus Networks

April 26, 2014

Eric Leblond a.k.a Regit

- French
- Network security expert
- Free Software enthousiast
- NuFW project creator (Now ufwi), EdenWall co-founder
- Netfilter developer:
  - Maintainer of ulogd2: Netfilter logging daemon
  - Misc contributions:
    - NFQUEUE library and associates
    - Port of some features iptables to nftables
- Currently:
  - co-founder of Stamus Networks, a company providing Suricata based network probe appliances.
  - Suricata IDS/IPS funded developer

What is Suricata

- IDS and IPS engine
- Get it here: `http://www.suricata-ids.org`
- Open Source (GPLv2)
- Funded by US government and consortium members
- Run by Open Information Security Foundation (OISF)
- More information about OISF at `http://www.openinfosecfoundation.org/`

Suricata Features

- High performance, scalable through multi threading
- Protocol identification
- File identification, extraction, on the fly MD5 calculation
- TLS handshake analysis, detect/prevent things like Diginotar
- Hardware acceleration support:
  - Endace
  - Napatech,
  - CUDA
  - PF_RING

Suricata Features

- Rules and outputs compatible to Snort syntax
- useful logging like HTTP request log, TLS certificate log, DNS logging
- Lua scripting for detection

Suricata capture modes

IDS

- pcap: multi OS capture
- pf_ring: Linux high performance
- af_packet: Linux high performance on vanilla kernel
- …

IPS

- NFQUEUE: Using Netfilter on Linux
- ipfw: Use divert socket on FreeBSD
- af_packet: Level 2 software bridge

Offline analysis

- Pcap: Analyse pcap files
- Unix socket: Use Suricata for fast batch processing of pcap files

Suricata 2.0 new features

- 'EVE' logging, our all JSON output for events: alerts, HTTP, DNS, SSH, TLS and (extracted) files
- much improved VLAN handling
- a detectionless 'NSM' runmode
- much improved CUDA performance

libhtp

- Security oriented HTTP parser
- Written by Ivan Ristić (ModSecurity, IronBee)
- Support of several keywords
  - http_method
  - http_uri & http_raw_uri
  - http_client_body & http_server_body
  - http_header & http_raw_header
  - http_cookie
  - serveral more…
- Able to decode gzip compressed flows

Using HTTP features in signature

Signature example: Chat facebook

```
alert http $HOME_NET any -> $EXTERNAL_NET any \
(
 msg:"ET CHAT Facebook Chat (send message)"; \
 flow:established,to_server; content:"POST"; http_method; \
 content:"/ajax/chat/send.php"; http_uri; content:"facebook.com"; http_header; \
 classtype:policy-violation; reference:url,doc.emergingthreats.net/2010784; \
 reference:url,www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/POLICY/POLICY_Facebook_Chat; \
 sid:2010784; rev:4; \
)
```

This signature tests:

- The HTTP method: *POST*
- The page: */ajax/chat/send.php*
- The domain: *facebook.com*

Extraction and inspection of files

- Get files from HTTP downloads and uploads
- Detect information about the file using libmagic
  - Type of file
  - Other details
  - Author (if available)
- A dedicated extension of signature language
- SMTP support coming soon

Dedicated keywords

- *filemagic* : description of content

```
alert http any any -> any any (msg:"windows exec"; \
                               filemagic:"executable for MS Windows"; sid:1; rev:1;)
```

- *filestore* : store file for inspection

```
alert http any any -> any any (msg:"windows exec";
                               filemagic:"executable for MS Windows"; \
                               filestore; sid:1; rev:1;)
```

- *fileext* : file extension

```
alert http any any -> any any (msg:"jpg claimed, but not jpg file"; \
                               fileext:"jpg"; \
                               filemagic:!"JPEG image data"; sid:1; rev:1;)
```

- *filename* : file name

```
alert http any any -> any any (msg:"sensitive file leak";
                               filename:"secret"; sid:1; rev:1;)
```

Examples

- Files sending on a server only accepting PDF

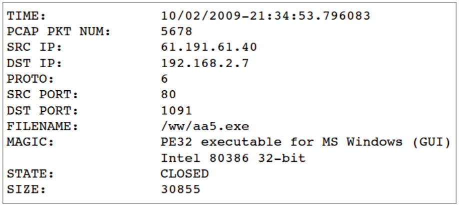
```
alert http $EXTERNAL_NET -> $WEBSERVER any (msg:"suspicious upload"; \
        flow:established,to_server; content:"POST" http_method; \
        content:"/upload.php"; http_uri; \
        filemagic:!"PDF document"; \
        filestore; sid:1; rev:1;)
```

- Private keys in the wild

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"outgoing private key"; \
        filemagic:"RSA private key"; sid:1; rev:1;)
```

Disk storage

- Every file can be stored to disk
- with a metadata file



```
TIME:              10/02/2009-21:34:53.796083
PCAP PKT NUM:      5678
SRC IP:            61.191.61.40
DST IP:            192.168.2.7
PROTO:             6
SRC PORT:          80
DST PORT:          1091
FILENAME:          /ww/aa5.exe
MAGIC:             PE32 executable for MS Windows (GUI)
                   Intel 80386 32-bit
STATE:             CLOSED
SIZE:              30855
```

- Disk usage limit can be set
- Scripts for looking up files / file md5's at Virus Total and others

A TLS handshake parser

- No traffic decryption
- Method
  - Analyse of TLS handshake
  - Parsing of TLS messages
- A security-oriented parser
  - Coded from scratch
    - Provide a hackable code-base for the feature
    - No external dependency (OpenSSL or GNUtls)
  - Contributed by Pierre Chifflier (ANSSI)
  - With security in mind:
    - Resistance to attacks (audit, fuzzing)
    - Anomaly detection

A handshake parser

- The syntax

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 443
```

- becomes

```
alert tls $HOME_NET any -> $EXTERNAL_NET any
```

- Interest:
  - No dependency to IP params
  - Pattern matching is limited to identified protocol
    - Less false positive
    - More performance

TLS keywords

- *tls.version*: Match protocol version number
- *tls.subject*: Match certificate subject
- *tls.issuerdn*: Match the name of the CA which has signed the key
- *tls.fingerprint*: Match the fingerprint of the certificate
- *tls.store*: Store certificates chain and a meta file on disk

Example: verify security policy (1/2)

- Environnement:
  - A company with servers
  - With an official PKI
- The goal:
  - Verify that the PKI is used
  - Without working too much

Example: verify security policy (2/2)

- Let’s check that the certificates used when a client negotiate a connection to one of our servers are the good one
- The signature:

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

Luajit rules

- Rule language is really simple
- Some tests are really difficult to write
    - Logic can be obtained via flow counters (flowbit) usage
    - But numerous rules are necessary
- A true language can permit to
    - Simplify some things
    - Realize new things

Experimental rules: `https://github.com/EmergingThreats/et-luajit-scripts`

Lua

Declaring a rule

```
alert tcp any any -> any any (msg:"Lua rule"; luajit:test.lua; sid:1;)
```

An example script

```
function init (args)
    local needs = {}
    needs["http.request_line"] = tostring(true)
    return needs
end
-- match if packet and payload both contain HTTP
function match(args)
    a = tostring(args["http.request_line"])
    if #a > 0 then
        if a:find("^POST%s+/.*%.php%s+HTTP/1.0$") then
            return 1
        end
    end
    return 0
end
```

heartbleed

The challenge

- No parsing of heartbeat, so hard solution
- Need pattern matching
- Easy to escape

Poor man solution

```
alert tcp any any -> any $TLS_PORTS (content:"|18 03 02|"; depth: 3; \
  content:"|01|"; distance: 2; within: 1;content:!"|00|"; within: 1; \
  msg: "TLSv1.1 Malicious Heartbleed RequestV2"; sid: 3;)
```

heartbleed

luajit to the rescue

- Heartbeat parameters are in clear (message type and length)
- Parsing of heartbeat messages can be done in luajit

```
alert tls any any -> any any ( \
    msg:"TLS HEARTBLEED malformed heartbeat record"; \
    flow:established,to_server; dsize:>7; \
    content:"|18 03|"; depth:2; lua:tls-heartbleed.lua; \
    classtype:misc-attack; sid:3000001; rev:1;)
```

heartbleed: the luajit script

```lua
function init (args)
    local needs = {}
    needs["payload"] = tostring(true)
    return needs
end

function match(args)
    local p = args['payload']
    if p == nil then
        --print ("no payload")
        return 0
    end

    if #p < 8 then
        --print ("payload too small")
        return 0
    end
    if (p:byte(1) ~= 24) then
        --print ("not a heartbeat")
        return 0
    end
```

```lua
    -- message length
    len = 256 * p:byte(4) + p:byte(5)
    --print (len)

    -- heartbeat length
    hb_len = 256 * p:byte(7) + p:byte(8)

    -- 1+2+16
    if (1+2+16) >= len  then
        print ("invalid length heartbeat")
        return 1
    end

    -- 1 + 2 + payload + 16
    if (1 + 2 + hb_len + 16) > len then
        print ("heartbleed detected: " \
.. (1 + 2 + hb_len + 16) .. " > " .. len)
        return 1
    end
    --print ("no problems")
    return 0
end
return 0
```

heartbleed: detection via the TLS parser

Using anomaly detection

- Decode protocol to fight evasion
- Available in suricata git 2 days after heartbleed and will be part of 2.0.1 (planned at beginning of May 2014)

The rules

```
alert tls any any -> any any ( \
   msg:"SURICATA TLS overflow heartbeat encountered, possible exploit attempt (heartbleed)"; \
   flow:established; app-layer-event:tls.overflow_heartbeat_message; \
   flowint:tls.anomaly.count,+,1; classtype:protocol-command-decode; \
   reference:cve,2014-0160; sid:2230012; rev:1;)
alert tls any any -> any any ( \
   msg:"SURICATA TLS invalid heartbeat encountered, possible exploit attempt (heartbleed)"; \
   flow:established; app-layer-event:tls.invalid_heartbeat_message; \
   flowint:tls.anomaly.count,+,1; classtype:protocol-command-decode; \
   reference:cve,2014-0160; sid:2230013; rev:1;)
```

More info on Victor Julien's blog

http://blog.inliniac.net/2014/04/08/
detecting-openssl-heartbleed-with-suricata/

Let’s get rid of the 90’s

Let’s kill unified2

- Binary format without real design
- Dedicated to alert
- Very hard to extend
- No API on devel side

We need something extensible

- To log alert and to log protocol request
- Easy to generate and easy to parse
- Extensible

JavaScript Object Notation

JSON

- JSON (`http://www.json.org/`) is a lightweight data-interchange format.
- It is easy for humans to read and write.
- It is easy for machines to parse and generate.
- An object is an unordered set of name/value pairs.

Logging in JSON

```
{"timestamp":"2012-02-05T15:55:06.661269", "src_ip":"173.194.34.51",
 "dest_ip":"192.168.1.22",
 "alert":{"action":"allowed",rev":1,"signature":"SURICATA TLS store"}}
```

Alert

The structure

- IP information are identical for all events and alert
- Follow Common Information Model
- Allow basic aggregation for all Suricata events and external sources

Example

```
{"timestamp":"2014-03-06T05:46:31.170567","event_type":"alert",
 "src_ip":"61.174.51.224","src_port":2555,
 "dest_ip":"192.168.1.129","dest_port":22,"proto":"TCP",
 "alert":{"action":"Pass","gid":1,"signature_id":2006435,"rev":8,
          "signature":"ET SCAN LibSSH Based SSH Connection - Often used as
          "category":"Misc activity","severity":3}
}
```

Network Security Monitoring

Protocols

- HTTP
- File
- TLS
- SSH
- DNS

Example

```
{"timestamp":"2014-04-10T13:26:05.500472","event_type":"ssh",
 "src_ip":"192.168.1.129","src_port":45005,
 "dest_ip":"192.30.252.129","dest_port":22,"proto":"TCP",
 "ssh":{
  "client":{
    "proto_version":"2.0","software_version":"OpenSSH_6.6p1 Debian-2" },
  "server":{
    "proto_version":"2.0","software_version":"libssh-0.6.3"}
 }
}
```

At the beginning was syslog

Pre Netfilter days

- Flat packet logging
- One line per packet
  - A lot of information
  - Non searchable

Not sexy

```
INPUT DROP IN=eth0 OUT= MAC=00:1a:92:05:ee:68:00:b0:8e:83:3b:f0:08:00 SRC=62.212.121.211 DST=91.12
IN IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=
IN IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.23 DST=192.168.11.3 LEN
IN IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=
IN IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=
```

Ulogd2: complete Netfilter logging

Ulogd2

- Interact with the post 2.6.14 libraries
- Rewrite of ulogd
- SCTP support (developed during @philpraxis talk at hack.lu 2008)
- multiple output and input through the use of stack

libnetfilter_log (generalized ulog)

- Packet logging
- IPv6 ready
- Few structural modification

libnetfilter_conntrack (new)

- Connection tracking logging
- Accounting, logging

Ulogd: output and configuration

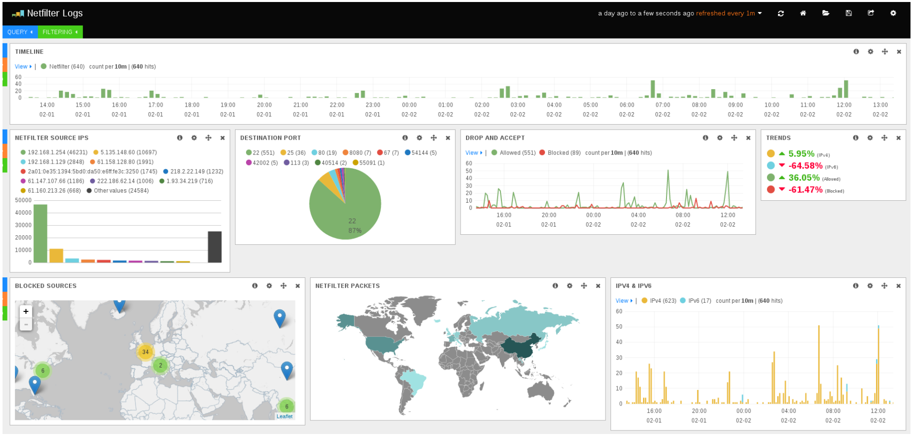
Sexify output

- Syslog and file output
- SQL output: PGSQL, MySQL, SQLite
- Graphite
- JSON output

Some stack examples

```
stack=log2:NFLOG,base1:BASE,ifi1:IFINDEX, \
      ip2str1:IP2STR,mac2str1:HWHDR,json1:JSON
stack=ct1:NFCT,mark1:MARK,ip2str1:IP2STR,pgsql2:PGSQL
```

Ulogd


Netfilter Logs
a day ago to a few seconds ago refreshed every 1m
QUERY
FILTERING
TIMELINE
View | Netfilter (640) count per 10m | (640 hits)
NETFILTER SOURCE IPS
192.168.1.254 (46231) 5.135.148.60 (10697)
192.168.1.129 (2848) 61.158.128.80 (1991)
2a01:0e35:1394:5bd0:da50:e6ff:fe3c:3250 (1745) 218.2.22.149 (1232)
61.147.107.66 (1186) 222.186.62.14 (1006) 1.93.34.219 (716)
61.160.213.26 (668) Other values (24584)
DESTINATION PORT
22 (551) 25 (36) 80 (19) 8080 (7) 67 (7) 54144 (5)
42002 (5) 113 (3) 40514 (2) 55091 (1)
22
87%
DROP AND ACCEPT
View | Allowed (551) Blocked (89) count per 10m | (640 hits)
TRENDS
5.95% (IPv4)
-64.58% (IPv6)
36.05% (Allowed)
-61.47% (Blocked)
BLOCKED SOURCES
34
2
6
6
Leaflet
NETFILTER PACKETS
IPV4 & IPV6
View | IPv4 (623) IPv6 (17) count per 10m | (640 hits)

ELK

- Elasticsearch is a distributed restful search and analytics
- Full text search, schema free
- Apache 2 open source license
- ELK stack
    - Elasticsearch
    - Logstash: log shipping
    - Kibana: web interface

Logstash

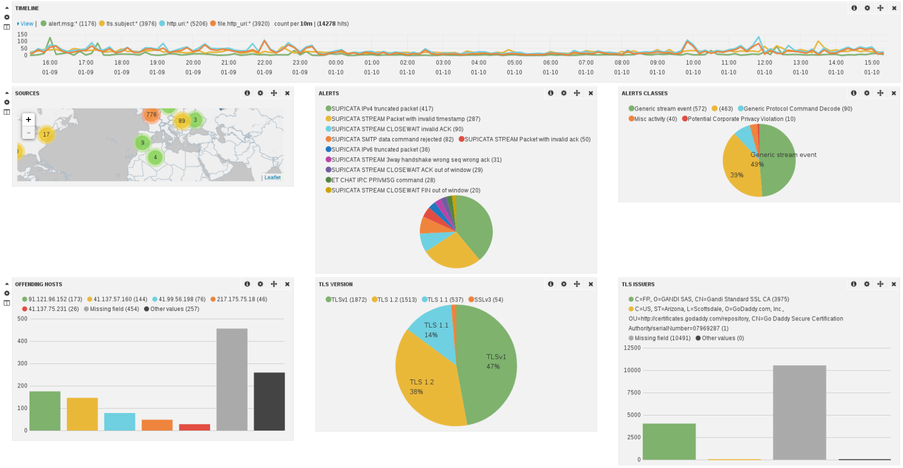
A tool for managing events and logs

- collect logs, parse them, and store them in different outputs
  - elasticsearch
  - graphite
  - IRC
  - ...
- Apache 2.0 license
-

A simple configuration (for JSON)

```
input {
   file {
      path => [ "/var/log/suricata/eve.json", "/var/log/ulogd.json"]
      codec =>   json
   }
}
```

Kibana


TIMELINE
View | ● alert.msg:* (1176) ● tls.subject:* (3976) ● http.uri:* (5206) ● file.http_uri:* (3920) count per 10m | (14278 hits)
150
100
50
0
16:00 01-09
17:00 01-09
18:00 01-09
19:00 01-09
20:00 01-09
21:00 01-09
22:00 01-09
23:00 01-09
00:00 01-10
01:00 01-10
02:00 01-10
03:00 01-10
04:00 01-10
05:00 01-10
06:00 01-10
07:00 01-10
08:00 01-10
09:00 01-10
10:00 01-10
11:00 01-10
12:00 01-10
13:00 01-10
14:00 01-10
15:00 01-10
SOURCES
776
89
3
17
9
4
Leaflet
ALERTS
● SURICATA IPv4 truncated packet (417)
● SURICATA STREAM Packet with invalid timestamp (287)
● SURICATA STREAM CLOSEWAIT invalid ACK (90)
● SURICATA SMTP data command rejected (82) ● SURICATA STREAM Packet with invalid ack (50)
● SURICATA IPv6 truncated packet (36)
● SURICATA STREAM 3way handshake wrong seq wrong ack (31)
● SURICATA STREAM CLOSEWAIT ACK out of window (29)
● ET CHAT IRC PRIVMSG command (28)
● SURICATA STREAM CLOSEWAIT FIN out of window (20)
ALERTS CLASSES
● Generic stream event (572) ● (463) ● Generic Protocol Command Decode (90)
● Misc activity (40) ● Potential Corporate Privacy Violation (10)
Generic stream event
49%
39%
OFFENDING HOSTS
● 91.121.96.152 (173) ● 41.137.57.160 (144) ● 41.99.56.198 (76) ● 217.175.75.18 (46)
● 41.137.75.231 (26) ● Missing field (454) ● Other values (257)
500
400
300
200
100
0
TLS VERSION
● TLSv1 (1872) ● TLS 1.2 (1513) ● TLS 1.1 (537) ● SSLv3 (54)
TLS 1.1
14%
TLSv1
47%
TLS 1.2
38%
TLS ISSUERS
● C=FR, O=GANDI SAS, CN=Gandi Standard SSL CA (3975)
● C=US, ST=Arizona, L=Scottsdale, O=GoDaddy.com, Inc., OU=http://certificates.godaddy.com/repository, CN=Go Daddy Secure Certification Authority/serialNumber=07969287 (1)
● Missing field (10491) ● Other values (0)
12500
10000
7500
5000
2500
0

STAMVS NETWORKS

Plotting TCP window at start

OS passive fingerprinting

- Value of TCP window at start is not specified in RFC
- The value is a choice of the OS
- We can use this for identification

Value for some OSes

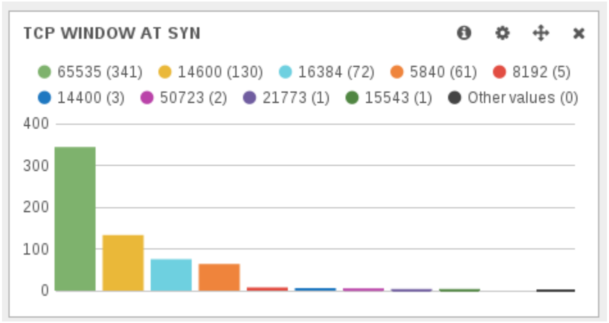
- 8192: Windows 7 SP1
- 65535: Mac OS X 10.2 - 10.7
- 14600: Some Linux
- 5840: Some other Linux

Source: `http://noc.to/#Help:TcpSynPacketSignature`

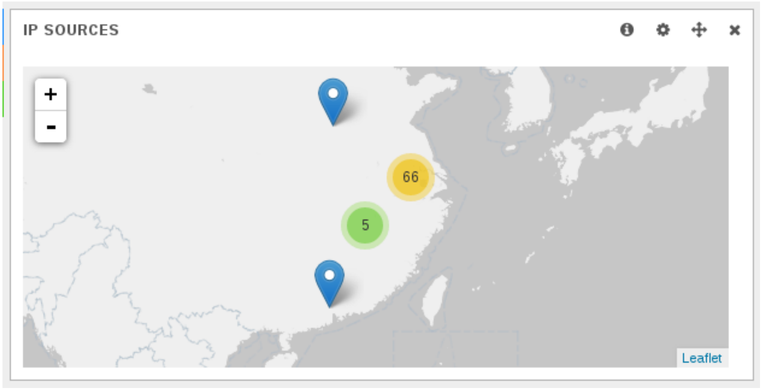
Demonstration

Let’s pray Murphy

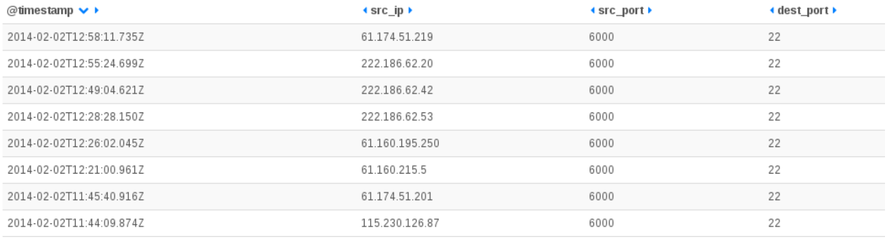
The facts


TCP WINDOW AT SYN
● 65535 (341)
● 14600 (130)
● 16384 (72)
● 5840 (61)
● 8192 (5)
● 14400 (3)
● 50723 (2)
● 21773 (1)
● 15543 (1)
● Other values (0)
400
300
200
100
0

The facts

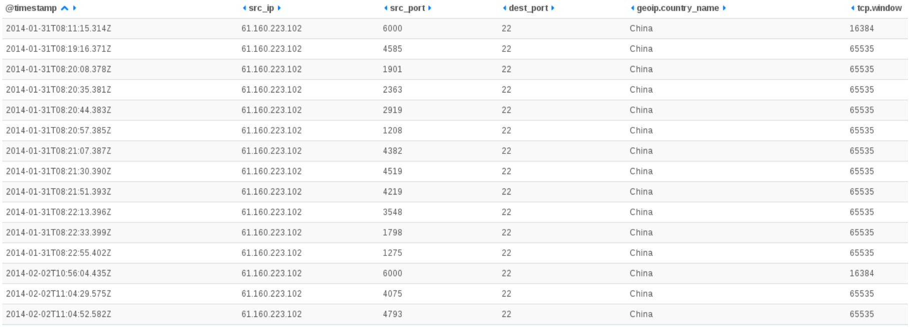


The facts



@timestamp	src_ip	src_port	dest_port
2014-02-02T12:58:11.735Z	61.174.51.219	6000	22
2014-02-02T12:55:24.699Z	222.186.62.20	6000	22
2014-02-02T12:49:04.621Z	222.186.62.42	6000	22
2014-02-02T12:28:28.150Z	222.186.62.53	6000	22
2014-02-02T12:26:02.045Z	61.160.195.250	6000	22
2014-02-02T12:21:00.961Z	61.160.215.5	6000	22
2014-02-02T11:45:40.916Z	61.174.51.201	6000	22
2014-02-02T11:44:09.874Z	115.230.126.87	6000	22

The facts



@timestamp	src_ip	src_port	dest_port	geoip.country_name	tcp.window
2014-01-31T08:11:15.314Z	61.160.223.102	6000	22	China	16384
2014-01-31T08:19:16.371Z	61.160.223.102	4585	22	China	65535
2014-01-31T08:20:08.378Z	61.160.223.102	1901	22	China	65535
2014-01-31T08:20:35.381Z	61.160.223.102	2363	22	China	65535
2014-01-31T08:20:44.383Z	61.160.223.102	2919	22	China	65535
2014-01-31T08:20:57.385Z	61.160.223.102	1208	22	China	65535
2014-01-31T08:21:07.387Z	61.160.223.102	4382	22	China	65535
2014-01-31T08:21:30.390Z	61.160.223.102	4519	22	China	65535
2014-01-31T08:21:51.393Z	61.160.223.102	4219	22	China	65535
2014-01-31T08:22:13.396Z	61.160.223.102	3548	22	China	65535
2014-01-31T08:22:33.399Z	61.160.223.102	1798	22	China	65535
2014-01-31T08:22:55.402Z	61.160.223.102	1275	22	China	65535
2014-02-02T10:56:04.435Z	61.160.223.102	6000	22	China	16384
2014-02-02T11:04:29.575Z	61.160.223.102	4075	22	China	65535
2014-02-02T11:04:52.582Z	61.160.223.102	4793	22	China	65535

Conclusion

Don’t fear to be sexy

- Sexy charts and interfaces are not only for finance guys thanks to Elasticsearch
- Suricata can boost the sex appeal of network monitoring

More information

- Suricata: `http://www.suricata-ids.org/`
- Netfilter: `http://www.netfilter.org/`
- Elasticsearch: `http://www.elasticsearch.org/`
- Suricata developers blogs: `http://planet.suricata-ids.org/`
- My blog: `https://home.regit.org/`
- Stamus Networks: `https://www.stamus-networks.com/`