Intrusion Detection and Suricata

Éric Leblond

Stamus Networks

February 9, 2015

IDS/IPS

Intrusion Detection System

> *An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. (source: Wikipedia)*

Two big families:

- NIDS: Network IDS
- HIDS: Host IDS

Intrusion Prevention System

> *Intrusion prevention systems (IPS), also known as intrusion detection and prevention systems (IDPS), are network security appliances that monitor network and/or system activities for malicious activity. (source: Wikipedia)*

HIDS

HIDS

- Monitor all aspects of operating system activity
- Method:
  - Maintain a database of items to monitor (storing checksum)
  - This can includes
    - specific parts of the memory
    - system call table on Linux, vtable on Windows
  - Detect if there is modifications (checksum change)

Solutions

- OSSEC
  - Open-source.
  - Run on Linux, MacOS, Solaris, HP-UX, AIX and Windows.
  - File Integrity checking, Log Monitoring, Rootkit detection
- Tripwire: A commercial HIDS
- Trusted Platform Module: an answer to the trust chain problem

IDS

Trafic analysis

- Use network view to find bad stuff in an enterprise network
- Detect:
  - Attack trafic: shell code
  - Network abuse: Insecure login
  - Post intrusion activity: command and control channel
  - Suspect behavior: abnormal network usage
  - Network security monitoring: keep a trace of event for forensics

Different technical approach

- Anomaly detection
- Content detection
- Trace keeping

Position of IDS in network

Where to put it?

- Need to receive all interesting traffic
- Noise may be an issue:
  - Getting the flow after firewall action
  - To only consider trafic reaching servers
  - Alternative can be to use two IDS

Which technology to use ?

- Port mirroring
  - Switches are able to copy all trafic to a specified port
  - Switched Port Analyzer (SPAN) on Cisco
  - Roving Analysis Port (RAP) on 3Com
- Network TAP can also be used

IPS

A firewall complement

- Need to be able to block packet
- Can be done via routing or bridging

A controversary system

- False positive conducts to disfunction
- A clever attacker can use feature to make a DOS
  - Send packets from spoofed IP
  - One packet attack is enough
  - And can causing network outage

Network Security Monitoring

Advanced attacks and APT

- Stealth method that can used advanced technics
- Often use 0-days
- Difficult to detect

Monitoring as a solution

- Store extensive information about network activity
- Can be used for forensics
- Some companies are selling storage system that allow you to replay one day or more of traffic
- Interesting products:
  - Bro IDS: `http://www.bro-ids.org/`
  - argus: `http://argus.tcp4me.com/`

Bro

A Network Security Monitoring software

- A comprehensive monitoring tool:
  - Extract indicator of network usage
  - With an In-Depth comprehension of protocols
- Developped and used at start by universities but now used more globally.
- Available under BSD licence.

Main features

- Scripting: a domain-specific scripting language
- Forensics: provides a high-level archive of a network's activity.
- In-depth Analysis: analyzers for many protocols
- Highly Stateful: Bro keeps extensive application-layer state.
- Cluster mode: Bro achieve scalability via transparent clustering.

Example script: SSL certificate validation

```
export {
  redef record Info += {
    validation_status: string &log &optional; ## Result
  };
  global recently_validated_certs: table[string] of string = table()
   &read_expire=5mins &synchronized &redef; ## MD5 hash cache
}
event ssl_established(c: connection) &priority=3 {
  if ( c$ssl?$cert_hash && c$ssl$cert_hash in recently_validated_certs ) {
    c$ssl$validation_status = recently_validated_certs[c$ssl$cert_hash];
  } else {
    local result = x509_verify(c$ssl$cert, c$ssl$cert_chain, root_certs);
    c$ssl$validation_status = x509_err2str(result);
  }
  if ( c$ssl$validation_status != "ok" ) {
   NOTICE([$note=Invalid_Server_Cert, $msg=message,
       $sub=c$ssl$subject, $conn=c,
       $identifier=cat(c$id$resp_h,c$id$resp_p,
              c$ssl$validation_status,
              c$ssl$cert_hash)]);
  }
}
```

Some words about this script

An advanced scripting language

- Inclusion of existing ressources (ssl parsing, . . .)
- Access to detailed part of the protocol
- Global variable can be used:
  - Sharing value and optimisation are easy to do.
  - And variables are shared in a cluster!

Scripting is cool

- Easy to hack and customize
- Depends of root_certs definition
- It is possible to make an instance for each browser

Anomaly detection technologies

Principes

- Use heuristics and rules to qualify normal trafic
- Need to learn normal system activity:
  - Via artificial intelligence (including neural network)
  - Via mathematical model

Problems

- High false positive rate
- Ability to be fooled by a correctly delivered attack

Signature based IDS

Look for motif in trafic

- Use a set of rules (signatures) to detect malicious content
- Trigger alert

Snort and Suricata

- Two open-source implementation
- Suricata uses snort rules language (with extensions)

Signatures

```
alert tcp any any -> 192.168.1.0/24 21 (content: "USER root"; msg: "FTP root login";)
```

Action: alert / drop / pass IP parameters Motif Other parameters

Evasion technics

Fooling detection

- Get your activity unnoticed
- Complete you attacka and stay in place

Principe

- Signature-base IDS relay on packet content
- Modification of trafic could be used to avoid detection
- Without changing the impact of the attack

Attacking the IDS

Fragmentation and Small Packets

- Split content on multiple packets
- A per-packet view will never see search content

Using an IDS vulnerability

- Attacking IDS will bring it down
- Next attacks will be unnoticed

Use IDS inperfect implementation

Protocol Violations

- Service can tolerate error
- IDS can ignore messages because of error

Obfuscating attack payload

- Multiple representation exists for a query
- Not using the standard way can evade IDS
- In Unicode, same string can be written in multiple way.

Inserting Traffic at the IDS

Different traffic for IDS and for target

- Hide attack by injecting data seen by IDS and not by target
- Attack based on knownledge of target network:
  - Send trafic that will be rejected by an active equipment after IDS
  - Use firewall filtering policy
  - Use other methods

TTL attack

- Use IDS inperfect knowledge of network
- Send low TTL packet seen by IDS but not by target

Play on interpretation issue

OS-based evasion

- All OS do not react the same
  - RFC are incomplete. Improvisations have been made.
  - Variation of traffic for a same flow is possible
- Overlapping Fragments

Application-based evasion

- Different server can treat the same request differently.
- No web server are treating a twice used argument the same way.

Personnality

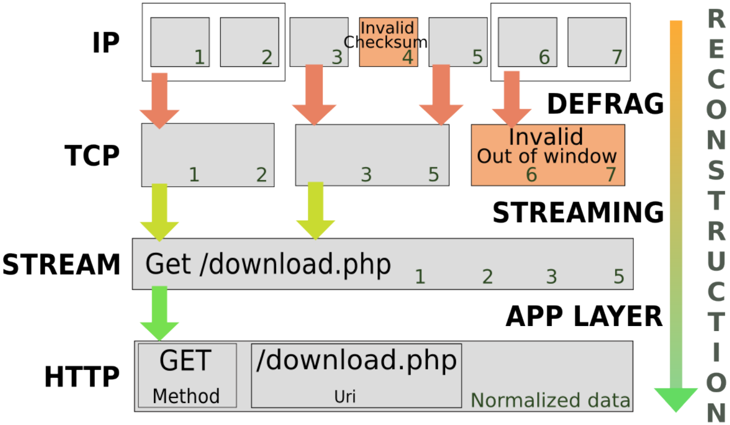
Personnality

- IDS implements personnality
- It is possible to associate network and OS type
- For Suricata, HTTP servers can be personnified too.

Suricata configuration

```
host-os-policy:
  # Make the default policy windows.
  windows: [0.0.0.0/0]
  bsd: []
  bsd-right: []
  old-linux: []
  linux: [10.0.0.0/8]
```

Suricata reconstruction and normalization



SIEM

Definition

Security Information and Event Management provides real-time analysis of security alerts generated by network hardware and applications. (source: Wikipedia)

Features

- Data Aggregation: get log from server and equipment, alerts from IDS
- Correlation: links event together, detect abnormal behavior
- Dashboards: generate charts using aggregated datas
- Retention: long-term storage to facilitate correlation and fullfill compliance requirements

Solutions

OSSIM

- Open Source: `http://communities.alienvault.com/`
- "Base" of commercial solution: `http://www.alienvault.com/`

HP ArcSight

- Commercial appliance-based solution

Prelude

- IDMEF implementation: http://www.ietf.org/rfc/rfc4765.txt
  - Normalisation of events
  - In an extensible XML based format
- `https://www.prelude-ids.org/`

About OISF

Open Information Security Foundation

- `http://www.openinfosecfoundation.org`
- Non-profit foundation organized to build a next generation IDS/IPS engine
- Funded by US Governement (DHS, Navy)
- Development of an Open Source IDS/IPS:
  - Developers financement
  - Financial support of related projects (barnyard2)
  - Board who defines big orientation
  - Roadmap is defined in public reunion

OISF Open Information Security Foundation

About OISF

- Consortium members
  - HOST program: Homeland Open Security Technology
  - Platinium level: BAE systems
  - Gold level: Npulse, Endace, Emerging Threats
  - Bronze level: SRC, Everis, Bivio networks, Nitro Security, Mara systems, . . .
  - Technology partner: Napatech, Nvidia
- Developers
  - Leader : Victor Julien
  - Developers: Anoop Saldanha, Gurvinder Singh, Pablo Rincon, William Metcalf, Eric Leblond, . . .
- Board
  - Matt Jonkmann
  - Richard Bejtlich, Dr. Jose Nazario, Joel Ebrahimi, Marc Norton, Stuart Wilson
  - . . .

Goals

- Bring new technologies to IDS
- Performance
  - Multi-threads
  - Hardware acceleration
  - `http://packetchaser.org/index.php/opensource/suricata-10gbps`
- Open source
- Support of Linux / *BSD / Mac OSX / Windows

Similar projects

Bro

- Different technology (capture oriented)
- Statistical study

Snort

- Equivalent
- Compatible
- Frontal concurrence

Suricata vs Snort

Suricata

- Drived by a foundation
- Multi-threaded
- Native IPS
- Advanced functions (flowint, libHTP)
- PF_RING support, CUDA support
- Modern and modular code
- Young but dynamic

Snort

- Developed by Sourcefire
- Multi-process
- IPS support
- SO ruleset (advanced logic + perf but closed)
- No hardware acceleration
- Old code
- 10 years of experience

Independant study:
`http://www.aldeid.com/index.php/Suricata-vs-snort`

Suricata with snort ruleset

- Not optimised
- Don’t use any advanced feature

Suricata with dedicated ruleset

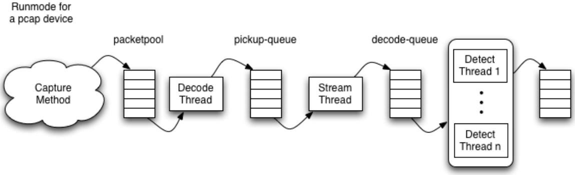
- Use Suricata optimised matchs
- Use Suricata advanced keywords
- Can get one from `http://www.emergingthreats.net/`

Fonctionnalities

- Ipv6 native support
- Multi-threaded
- Native hardware acceleration (PF_RING, Napatech, Endace, Myricom)
- Numerous options for performance optimisation
- Optimized support of IP only tests
- IPS is native (inline mode)
- Protocol detection

Global architecture

- Chained treatment modules
- Each *running mode* can have its own architecture
- Architecture of mode "pcap auto v1":



- Fine setting of CPU preferences
  - Attach a thread to a CPU
  - Attach a threads family to a CPU set
  - Allow IRQs based optimisation

Entry modules

IDS

- PCAP
  - live, multi interface
  - offline support
- AF_PACKET
- PF_RING: mutltithread
  `http://www.ntop.org/PF_RING.html`
- Capture card support: Napatech, Myricom, Endace

IPS

- NFQueue:
  - Linux: multi-queue, advanced support
  - Windows
- ipfw :
  - FreeBSD
  - NetBSD

Output modules

- Fastlog
- Unified log (Barnyard 1 & 2)
- HTTP log (log in apache-style format)
- Prelude (IDMEF)

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

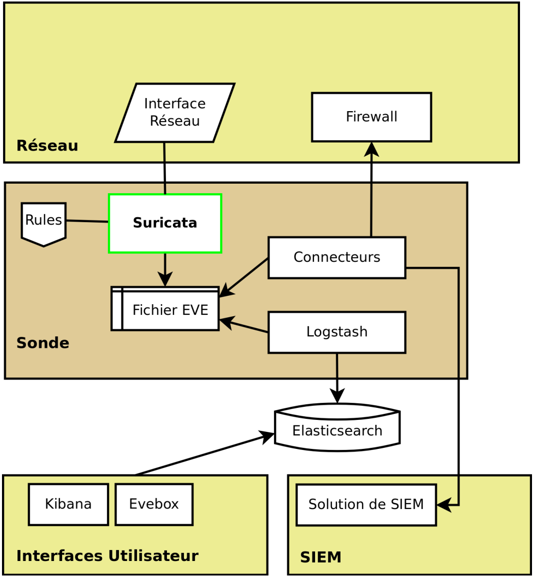
Example

```
{"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"}
 }
}
```

Output modules

- EVE format
- Fastlog
- Unified log (Barnyard 1 & 2)
- HTTP log (log in apache-style format)
- Prelude (IDMEF)

Suricata Ecosystem



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

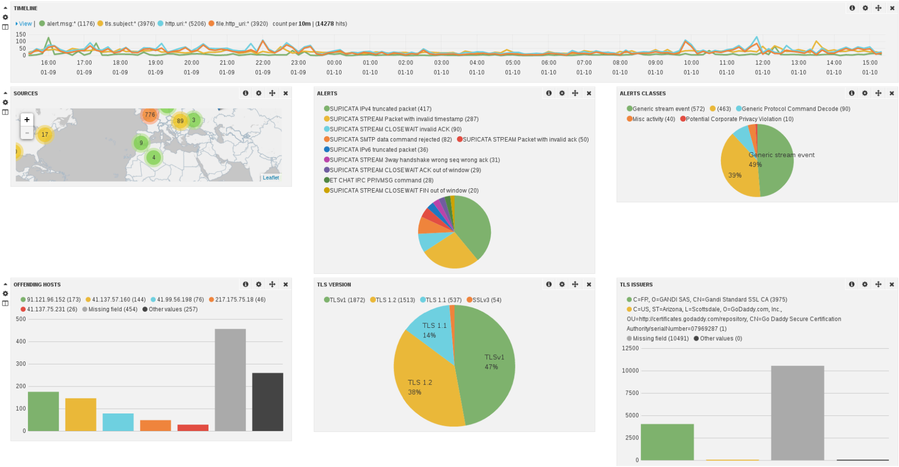
A tool for managing events and logs

- 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

Stream inline

- High level applicative analysis works on a data stream
- TCP data can be messy
  - Packets loss
  - Packets retransmit
  - Out of order packets
- The $I_P^D S$ must reconstruct the TCP flow before doing the applicative analysis

Problem

- IDS must be the closer possible to what's received by the target
    - Packet analysis when reception has been proven
    - ACK reception trigger data analysis
- IPS must block the packets before they reached the target
    - The IDS algorithm will block packet *after* they go through
    - An other approach has to be used

IPS as a control point

- IPS is a blocking point
  - It is representative of what goes through
  - It can reconstruct the flows before send them
- Suricata implementation
  - Reconstruction of data segments at reception
  - Send reconstructed data to applicative layer analyser
  - Take decision based on data
  - Rewrite packets if necessary
  - Transmit (possibly modified) packets
- Details: `http://www.inliniac.net/blog/2011/01/31/suricata-ips-improvements.html`

libHTP

- Security oriented HTTP parser
- Written by Ivan Ristić (ModSecurity, IronBee)
- Flow tracking
- Support of keywords
  - http_body
  - http_raw_uri
  - http_header
  - http_cookie
  - …
- Able to decode gzip compressed flows

Using HTTP features in signature

Signature example: Chat facebook

```
alert http $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS \
(
 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*

Flow variables

Objectives

- Detection of in-multiple-step attack
- Verify condition on a flow
- Modify alert treatment
- State machine inside each flow

Flowbits

- boolean condition
- Set a flag

Flowint

- Define counter
- Arithmetic operation

Extraction et inspection of files

- Get files from HTTP and SMTP downloads and uploads
- Detect information about the file using libmagic
  - Type of file
  - Other details
  - …
- A dedicated extension of signature language

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

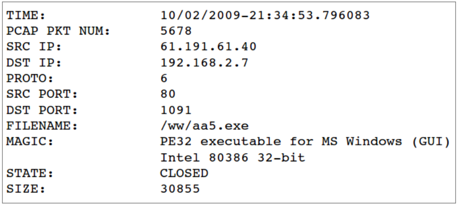
```
alert http $EXTERNAL_NET -> $WEBSERVER any (msg:"suspicious upload"; \
        flow:established,to_server; content:"POST" http_method; \
        content:"/upload.php"; http_uri; \
        filemagic:!"PDF document"; \
        filestore; sid:1; rev:1;)
```

- 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 is stored on 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

Actual limit of files extraction

- Limited to the HTTP and SMTP protocol
- Storage limit are suboptimal
- MS Office files are not decoded

TLS Handshake parser

- TLS is an application in Suricata way
- Automatic detection of protocol
  - Independent of port
  - Made by pattern matching
- Dedicated keywords
- Usable in the signatures

Other supported applications

- *HTTP* :
  - keywords: http_uri, http_body, http_user_agent, . . .
- *SMTP*
- *FTP*
  - keyword: ftpbounce
- *SSH*
  - keywords: ssh.softwareversion, ssh.protoversion
- *DCERPC*
- *SMB*
- *Modbus*
  - keywords: function, subfunction, address

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
  - 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
- More to come

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";)
```

Example: detect certificate anomaly

- Google.com is signed by Google Internet Authority
- Not by an other CA (Diginotar by example)
- If it is the case, this is bad!
- Let's block that!

- Signature:

```
drop tls $CLIENT any -> any any ( \
  tls.subject:"C=US, ST=California, L=Mountain View, O=Google Inc, CN=*.google.com"; \
  tls.issuerdn:!"C=US, O=Google Inc, CN=Google Internet Authority";)
```

- What! KPN has been hacked too!
- Let's get rid of the Dutch!

```
drop tls $CLIENT any -> any any (tls.issuerdn:"C=NL");
```

Actual limit

- Keywords apply only to first certificate of the chain.
  - Impossible to do check on chained certificates
  - Supported by the parser but not by the keywords.
- Some keyword are missing and will be added
  - used cryptographic algorithm
  - Key size
  - Diffie-Hellman parameters
- Statistical study and certificate storage

Règles luajit

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

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
```

Signatures

```
alert tcp any any -> 192.168.1.0/24 21 (content: "USER root"; msg: "FTP root login";)
```

Action: alert / drop / pass IP parameters Motif Other parameters

Keywords documentation

Available on OISF wiki: `https://redmine.openinfosecfoundation.org/projects/suricata/wiki/`

Keywords description

Listing

```
# suricata —list–keywords
– filename
– luajit
– iprep
```

Export CSV

```
# suricata —list–keywords=cvs>keywords.csv
```

Keyword detail

```
# suricata —list–keywords=filename
= filename =
Description: match on the file name
Protocol: http
Features: none
Documentation: https://redmine.openinfosecfoundation.org/projects/suricata
```

Basic recommandations

Analyse resistance of match

- Can the motif be changed without behavior change ?
- Consider working on normalized protocol

Take care of performance

- Avoid regular expression
- Check performance
  - Rules analysis with `suricata -engine-analysis`

Performance analysis

```
== Sid: 2012233 ==
alert http $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"ET ACTIVEX Oracle Document Capture File
    Rule matches on reassembled stream.
    App layer protocol is ALPROTO_HTTP.
    Rule contains 2 content options, 0 http content options, 1 pcre options, and 0 pcre options wi
    Fast Pattern "4932CEF4-2CAA-11D2-A165-0060081C43D9" on "reassembled stream" buffer.
    Warning: Rule app layer protocol is http, but pcre options do not have http modifiers.
             -Consider adding http pcre modifiers.
    Warning: Rule app layer protocol is http, but content options do not have http_* modifiers.
             -Consider adding http content modifiers.
    Warning: Rule app layer protocol is http, but the fast_pattern is set on the raw stream.  Cons
```

Rules profiling

Special build is needed

- Add `-enable-profiling` to configure options.
- Check `profiling` section in the YAML configuration file.

Extract of rule_perf.log

```
----------------------------------------------------
Date: 1/8/2013 -- 15:06:36
----------------------------------------------------
 Rule    Gid Rev Ticks    %    Checks Matches  Max Ticks Avg Ticks Avg Match Avg No Match
-------- --- --- -------- ---- ------ -------- --------- --------- --------- ------------
2014894  1   4   228715   0.00 1      0        228715    228715.00 0.00      228715.00
2002029  1   11  3540157  0.01 39     0        276044    90773.26  0.00      90773.26
2006385  1   9   41521    0.00 1      0        41521     41521.00  0.00      41521.00
```

- Ticks: interval between to timer interrupt (4ms on test system)
- Match and No Match must be considered
- Total ticks give global impact

Analysing the worst rule

Here's the criminal

```
alert http $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"ET CURRENT_EVENTS RedKit - Landing Page Received - applet and 5dig
 flow:established,to_client; content:"<applet"; fast_pattern;
 content:".jar"; distance:0; \
 pcre:"/\W[0-9]{5}\.jar/"; classtype:trojan-activity; \
 sid:2014894; rev:4;)
```

Guilty of

- Regular expression usage
  - With a word search
- Matching on raw data for an http rule:

```
Rule app layer protocol is http, but the fast_pattern is set on
the raw stream.  Consider adding fast_pattern over a http buffer
for increased performance.
```

Fixing the signature

Adding http context

```
alert http $EXTERNAL_NET any -> $HOME_NET any \
 (msg:"ET CURRENT_EVENTS RedKit - Landing Page Received - applet and 5dig
 flow:established,to_client; \
 content:"<applet"; http_server_body; fast_pattern; \
 content:".jar";  http_server_body; distance:0; \
 pcre:"/\W[0-9]{5}\.jar/";
 classtype:trojan-activity; sid:2014894; rev:5;)
```

Results

- Max Ticks get from 228715 to 19950