A short introduction to Suricata \( IP^S \)

Éric Leblond

OISF

July 12th 2011
Introduction
- Introduction
- Goals of the project
- Ecosystem

Functionnalities
- List of functionnalities
- Signatures
- Stream inline
- CUDA

Advanced functionalities of Suricata
- libHTP
- Flow variables
- IPS advanced functions

The future
- The roadmap
- More information
(C) Jean-Marie Hullot, CC BY 3.0
A short introduction to Suricata

(C) Jean-Marie Hullot, CC BY 3.0
Éric Leblond

- Initial and lead developer of NuFW
- Netfilter Contributor (mainly ulogd2 and userpace interaction)
- Suricata core developer (IPS, multicore optimisation, . . .)
- Independant Open Source et security consultant
- . . .
Open Information Security Foundation

- [http://www.openinfosecfoundation.org](http://www.openinfosecfoundation.org)
- Non-profit foundation organized to build a next generation IDS/IPS engine
- Funded by US Government (DHS, Navy)
- Development of an Open Source IDS/IPS:
Open Information Security Foundation

- http://www.openinfosecfoundation.org
- Non-profit foundation organized to build a next generation IDS/IPS engine
- Funded by US Government (DHS, Navy)
- Development of an Open Source IDS/IPS:
  - Developers financement
Open Information Security Foundation

- [http://www.openinfosecfoundation.org](http://www.openinfosecfoundation.org)
- Non-profit foundation organized to build a next generation IDS/IPS engine
- Funded by US Government (DHS, Navy)
- Development of an Open Source IDS/IPS:
  - Developers financement
  - Financial support of related projects (barnyard2)
About OISF

Open Information Security Foundation

- [http://www.openinfosecfoundation.org](http://www.openinfosecfoundation.org)
- Non-profit foundation organized to build a next generation IDS/IPS engine
- Funded by US Government (DHS, Navy)
- Development of an Open Source IDS/IPS:
  - Developers financement
  - Financial support of related projects (barnyard2)
  - Board who defines the roadmap
Consortium members

- HOST program: Homeland Open Security Technology
- Gold level: Npulse, Endace
- Bronze level: EdenWall, Nitro Security, Mara systems, . . .
Consortium members

- HOST program: Homeland Open Security Technology
- Gold level: Npulse, Endace
- Bronze level: EdenWall, Nitro Security, Mara systems, . . .
- Technology partner: Napatech, Nvidia
Consortium members

- HOST program: Homeland Open Security Technology
- Gold level: Npulse, Endace
- Bronze level: EdenWall, Nitro Security, Mara systems, ...
- Technology partner: Napatech, Nvidia

Developers

- Leader: Victor Julien
About OISF

- Consortium members
  - HOST program: Homeland Open Security Technology
  - Gold level: Npulse, Endace
  - Bronze level: EdenWall, Nitro Security, Mara systems, . . .
  - Technology partner: Napatech, Nvidia

- Developers
  - Leader: Victor Julien
About OISF

- Consortium members
  - HOST program: Homeland Open Security Technology
  - Gold level: Npulse, Endace
  - Bronze level: EdenWall, 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
  - ...

Éric Leblond (OISF)
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

Sourcefire has felt endangered and has been aggressive

Volume of code

**Suricata**

![Suricata Volume of Code Chart]

**Snort**

![Snort Volume of Code Chart]

Source: ohloh.net
## 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:**
Introduction
- Introduction
- Goals of the project
- Ecosystem

Functionnalities
- List of functionnalities
- Signatures
- Stream inline
- CUDA

Advanced functionalities of Suricata
- libHTP
- Flow variables
- IPS advanced functions

The future
- The roadmap
- More information
Fonctionnalités

- IPv6 native support
Fonctionnalités

- IPv6 native support
- Multi-threaded
- IPv6 native support
- Multi-threaded
- Native hardware acceleration (GPU, PF_RING)
• IPv6 native support
• Multi-threaded
• Native hardware acceleration (GPU, PF_RING)
• Numerous options for performance optimisation
Fonctionnalités

- Ipv6 native support
- Multi-threaded
- Native hardware acceleration (GPU, PF_RING)
- Numerous options for performance optimisation
- Optimized support of IP only tests
Fonctionnalités

- IPv6 native support
- Multi-threaded
- Native hardware acceleration (GPU, PF_RING)
- Numerous options for performance optimisation
- Optimized support of IP only tests
- IPS is native (inline mode)
Global architecture

- Chained treatment modules
- Each *running mode* can have its own architecture
Global architecture

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

![Diagram of Suricata architecture](image)
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
### IDS

- **PCAP**
  - live, multi interface
  - offline support

- **PF_RING**
  - [http://www.ntop.org/PF_RING.html](http://www.ntop.org/PF_RING.html)
  - Multithread, really fast but require modified drivers
## Entry modules

### IDS

- **PCAP**
  - live, multi interface
  - offline support
- **PF_RING**
  - [http://www.ntop.org/PF_RING.html](http://www.ntop.org/PF_RING.html)
  - Multithread, really fast but require modified drivers

### 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)
Signatures

- Support almost all snort ruleset features
- Exclusive features used by VRT ou Emerging Threats rulesets

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

- Support almost all snort ruleset features
- Exclusive features used by VRT ou Emerging Threats rulesets

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

Action: alert / drop / pass
Signatures

- Support almost all snort ruleset features
- Exclusive features used by VRT ou Emerging Threats rulesets

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

IP parameters
Signatures

- Support almost all snort ruleset features
- Exclusive features used by VRT ou Emerging Threats rulesets

<table>
<thead>
<tr>
<th>Action: alert / drop / pass</th>
</tr>
</thead>
</table>

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

Motif
Signatures

- Support almost all snort ruleset features
- Exclusive features used by VRT ou Emerging Threats rulesets

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

Other parameters
High level applicative analysis works on a data stream

TCP data can be messy
- Packets loss
- Packets retransmit
- Out of order packets

The $I_P$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

Details: [http://www.inliniac.net/blog/2011/01/31/suricata-ips-improvements.html](http://www.inliniac.net/blog/2011/01/31/suricata-ips-improvements.html)
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

Éric Leblond (OISF)
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](http://www.inliniac.net/blog/2011/01/31/suricata-ips-improvements.html)
CUDA

- Offload some computation to GPU through CUDA which is a parallel computation library developed by NVIDIA
- Now: implementation of a matching algorithm in CUDA
- Work in progress, Nvidia is a technological partner of OISF
• Offload some computation to GPU through CUDA which is a parallel computation library developed by NVIDIA
• Now: implementation of a matching algorithm in CUDA
• Work in progress, Nvidia is a technological partner of OISF
• Difficult to use the GPU pipeline in an effective manner
Offload some computation to GPU through CUDA which is a parallel computation library developed by NVIDIA

Now: implementation of a matching algorithm in CUDA

Work in progress, Nvidia is a technological partner of OISF

Difficult to use the GPU pipeline in an effective manner

... Performance equivalent with and without CUDA (for decent CPUs)
Introduction
- Introduction
- Goals of the project
- Ecosystem

Functionnalities
- List of functionnalities
- Signatures
- Stream inline
- CUDA

Advanced functionalities of Suricata
- libHTP
- Flow variables
- IPS advanced functions

The future
- The roadmap
- More information
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
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
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
Flowint variables

- Permit capture, keep and comparison of data in one variable
- keep and do mathematical operations
- Variable is bound to a given flow

Ex: show an alert if and only if \texttt{usernamecount} is greater than 5:

\begin{verbatim}
alert tcp any any -> any any (msg: "Counting Usernames"; content: "jonkman"; \nflowint: usernamecount, +, 1; flowint: usernamecount, >, 5;)
\end{verbatim}
Flowint variables (2)

Ex: Follow logins
Put a login failure counter:

```
alert tcp any any -> any any (msg:"Start a login count"; content:"login failed"; \ 
flowint:loginfail, notset; flowint:loginfail, =, 1; flowint:noalert;)
alert tcp any any -> any any (msg:"Counting Logins"; content:"login failed"; \ 
flowint:loginfail, isset; flowint:loginfail, +, 1; flowint:noalert;)
```

Alert if there is a success after 5 failed login:

```
alert tcp any any -> any any (msg:"Login success after file failures"; \ 
content:"login successful"; \ 
flowint:loginfailed, isset; flowint:loginfailed, =, 5;)
```
Using a Linux/Netfilter based IPS

- Use NFQUEUE to send decision to userspace
- All packets of a connexion must be seen to Suricata
- The brutal way: `iptables -A FORWARD -j NFQUEUE`
Suricata in IPS mode

Using a Linux/Netfilter based IPS

- Use NFQUEUE to send decision to userspace
- All packets of a connexion must be seen to Suricata
- The brutal way: iptables -A FORWARD -j NFQUEUE

Interaction with the firewall

- NFQUEUE is a terminal target
  - An ACCEPT decision will shortcut the whole ruleset
  - This is the only possible decision but DROP
- The previous method is thus incompatible with the existence of a ruleset.
# Classic solution

Use mangle in the PREROUTING or FORWARD chains

- The rule is an isolated table
- Thus no interaction with the rest of the ruleset
- This mean we can do "nothing" in theses mangle chains

---

Living together: the IPS and the firewall case

Classic solution

Use mangle in the PREROUTING or FORWARD chains

- The rule is an isolated table
- Thus no interaction with the rest of the ruleset
- This mean we can do "nothing" in theses mangle chains

Alternative solution

- Use advanced functionalities of NFQUEUE
- Simulate a non terminal decision (© Patrick Mchardy)

Details: http://home.regit.org/2011/01/building-a-suricata-compliant-ruleset/
Alternate decisions

- **NF_REPEAT**: send the packet back to the start of the table
- **NF_QUEUE**: send the packet to another queue (chain software using NFQUEUE)

Alternate decisions

- NF_REPEAT : send the packet back to the start of the table
- NF_QUEUE : send the packet to another queue (chain software using NFQUEUE)

nfq_set_mark

- New keyword that can be used in signature
- Put a Netfilter mark on the packet if the signature match
- Can be used in every network stack (QoS, routing, Netfilter)

Objective

- Detect a suspect behaviour
- Increase logging for the whole connexion
Logging of a suspect connexion (1/2)

Objective
- Detect a suspect behaviour
- Increase logging for the whole connexion

Method
- The alert put a Netfilter mark on the packet
- Netfilter propagate the mark to all packets of the related connexion
- Netfilter log every marked packets
The alert in Suricata

```
pass tcp any any → any any (msg:"We were expecting you"; content:"Mr Bond"; \n nfq_set_mark:0x007/0xffff ;)
```
The alert in Suricata

```
pass tcp any any → any any (msg: "We were expecting you"; content: "Mr Bond"; \ nfq_set_mark:0x007/0xfff ;)
```

Netfilter settings

```
iptables -I PREROUTING -t mangle -j CONNMARK —restore—mark
iptables -A POSTROUTING -t mangle -j CONNMARK —save—mark
iptables -A POSTROUTING -t mangle -m mark —mark 0x007/0xfff —j NFLOG —nflog—prefix "Dr No log"
```
The alert in Suricata

```
pass tcp any any -> any any (msg: "We were expecting you"; content: "Mr Bond"; \n nfq_set_mark: 0x007/0 xfff ;)
```

Netfilter settings

```
iptables -I PREROUTING -t mangle -j CONNMARK —restore—mark
iptables -A POSTROUTING -t mangle -j CONNMARK —save—mark
iptables -A POSTROUTING -t mangle -m mark —mark 0x007/0 xfff —j NFLOG —nflog—prefix "Dr No log"
```

Next you can have ulogd2 to send everything in pcap ou SQL
1 Introduction
   • Introduction
   • Goals of the project
   • Ecosystem

2 Functionnalities
   • List of functionnalities
   • Signatures
   • Stream inline
   • CUDA

3 Advanced functionalities of Suricata
   • libHTP
   • Flow variables
   • IPS advanced functions

4 The future
   • The roadmap
   • More information
Roadmap

- Finalize CUDA acceleration
- IP and DNS reputation
- Extract files and inspect their content
- SCADA Preprocessor (thanks to Digital Bond)
- Keyword `replace`
- Keyword `geoip`
- Reload ruleset without breaking the flow analysis
- Stateful Pattern Matching/Transaction-Aware Detections

How to test it fast and easy?

- Already available in Debian, Ubuntu, Gentoo, FreeBSD
- Live distribution:

- Smooth-Sec (Suricata + Snorby): http://bailey.st/blog/smooth-sec/
Questions

Do you have questions?

- Big thanks:
  - Pierre Chifflier: [http://www.wzdftpd.net/blog/](http://www.wzdftpd.net/blog/)
  - The whole OISF team and especially Victor Julien

- Related read:
  - Suricata devel site: [https://redmine.openinfosecfoundation.org/](https://redmine.openinfosecfoundation.org/)
  - Victor Julien’s blog: [http://www.inliniac.net/blog/](http://www.inliniac.net/blog/)
  - Regit’s blog: [http://home.regit.org](http://home.regit.org)

- Join me:
  - Mail: eric@regit.org
  - Twitter: Regiteric