What is an IDS and Network Security Monitoring in 2023?

Suricata
Agenda

About us
What is Suricata
How it started
How it evolved
Challenges when monitoring traffic
How to get involved/contribute and stay in touch
Eric Leblond
CTO at Stamus Networks
OISF Team - Developer/Trainer
OISF Board of Directors
Linux Kernel/Netfilter
developer
Scirius CE/SELKS maintainer
@regit @regiteric
Peter Manev
@pevma
13 yrs with Suricata
OISF Exec team
Suricata QA/Training lead
CSO Stamus Networks
SELKS maintainer
Me likes -
Open Source
Threat Hunting
What is Suricata
What is Suricata?

- A high-performance network monitoring and security engine with active/passive monitoring, metadata logging and real-time file identification and extraction

- Powered by Open Source GPLv2 - find it on Github:  
  ○ https://github.com/OISF/suricata

- Produces a high-level of situational awareness and detailed application layer transaction records from network traffic.

- Used by thousands of organisations and ppl around the globe
What is Suricata?

Suricata can be deployed as

- **IDS** - Intrusion Detection System (passive sniffing)
- **IPS** - Intrusion Prevention system (inline)
- **NSM** - Network Security Monitoring (works without rules)
  - Protocol, flow and filetranscation logging
- **FPC** - Full Pcap Capture
  - Also possible: **Conditional** PCAP Capture
  - Thanks Eric Leblond!
- Combinations of the above like
  - IDS + NSM + FPC
  - IDS + Conditional PCAP capture
Suricata - Major Features

• Standards based formats (YAML, JSON) ease integrations with SIEM tools such as Elastic and Splunk

• Multithreaded, hardware acceleration available. 100Gb+ deployments

• Network metadata logging for a variety of protocols

• Advanced HTTP, DNS, SMTP, SMB and TLS support

• File identification and extraction - FTP/SMTP/HTTP/HTTP2/NFS/SMBv1-3

• Support for SCADA protocols - DNP3, ENIP, and CIP
Why The Network?

● The network is now the backbone of society
  ○ Connects computers for everything from social media to finance
● Criminals and other threat actors also utilize the network:
  ○ To attack the user
  ○ To deliver malware and other tools
  ○ To steal data
● Monitoring the network helps you to identify and stop this malicious activity
Network Metadata Logging

- Provides extensive logging of protocol and other network data
- Data logged in event records: HTTP/HTTP2, DNS, FTP, TLS, SMB, SSH, RDP...
- Default output format in JavaScript Object Notation (JSON)
File Identification and Extraction

- Can perform file identification and extraction in real-time

- File information includes:
  - Content type/libmagic
  - File hashes (MD5/SHA1/SHA2)
  - File size

- Files can also be extracted and stored to the file system
PCAP Capabilities

• Suricata can read PCAPs for offline processing
  • Ability to read a single PCAP or an entire directory
  • Can also process PCAPs through a Unix socket

• Suricata can also produce full packet capture (FPC)
  • Stored network data in PCAP files

• Consider multiple Suricata instances for testing/exploration/malware analysis
Passive Monitoring

Internet

Firewall

Router/Switch

Monitored Network

TAP/SPAN

Suricata

Alerts
Metadata
File information
PCAP

Monitored Network
Active Monitoring

Internet — Firewall — Suricata — Router/Switch

DROP/REJECT Traffic

Monitored Network

Monitored Network

Monitored Network

Alerts
Metadata
File information
PCAP
How Signatures Work

alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET INFO PS1 Powershell File Request"; flow:established,from_client; flowbits:set,ET-PS.Download; http.request_line; content:".*.ps1 HTTP/1."; nocase; fast_pattern; classtype:bad-unknown; sid:2032162; rev:1; metadata:affected_product Windows XP Vista 7 8 10 Server 32 64 Bit, attack.target Client_Endpoint, created at 2021_03_18, deployment Perimeter, former_category INFO, signature_severity Informational, updated at 2021_03_18;)

Malicious Document

Firewall
Suricata
Router/Switch
Employee
Network Request
Suricata History
Suricata History

● First lines of code written in 2007 by Victor Julien
  ○ First released in 2009

● Powered by Open Source GPLv2 (source on GitHub)

● Worked on/Developed with a global open source community in over 23 different countries

● Owned and supported by Open Information Security Foundation, a 501(c)3 non-profit
  ○ https://oisf.net
Suricata History

Brief History of Suricata

- 2009: OISF Founded & Suricata 1.0 Released
- 2010: First Suricata Code
- 2013: First Suricata Training
- 2015: Suricata Team Grows
- 2018: First SuriCon
- 2020: Suricata 6.0 Released
What is Suricata?

How it started?

- An example of how IDS alert looked back 14+ yrs ago

```
logs/fast.log
```
What is Suricata?

- **14 yrs** ago - You had to go deploy other tools to find the logs related to this event and figure out if it is TP or FP

```
```

```
```
What is Suricata? How it looks today?
What is Suricata? How it looks today?

**Signature**
- **Signature**: ET MALWARE Likely Malicious Windows SCT Download...
- **SID**: 2024902
- **Category**: A Network Trojan was detected
- **Severity**: Severe
- **Revision**: 2

**IP and basic information**
- **Source IP**: 104.21.78.47
- **Source port**: 80
- **Destination IP**: 10.1.5.101
- **Destination port**: 63285
- **IP protocol**: TCP
- **Application protocol**: HTTP
- **Probe**: 2023-01-05-Astaroth-Guildma-Infection-trafica.pcap

**Enrichment**
- **Source IP**: 104.21.78.47
- **Source port**: 80
- **Target IP**: 10.1.5.101
- **Target port**: 63285

**Flow**
- **Flow ID**: 772071985486292
- **Flow start**: 2023-01-05T02:32:10.114226-0000
- **Bytes to server**: 602
- **Packets to server**: 7
- **Bytes to client**: 4792
- **Packets to client**: 6

**Signature metadata**
- **attack_target**: Client_Endpoint
- **updated_at**: 2017_08_22
- **created_at**: 2017_08_22
- **signature_severity**: Major
- **deployment**: Perimeter
- **affected_product**: Windows_XP_Vista_7_8_10_Server_32_64_Bit
- **malware_family**: PowerShell_Downloader
- **performance_impact**: Low
- **former_category**: CURRENT_EVENTS
- **tag**: PowerShell
What is Suricata? How it looks today?

EveBox - Showcasing Flow ID
https://evebox.org/
### What is Suricata? How it looks today?

#### EveBox - Showcasing Flow ID

**Flow ID:** `772071985486292`

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Type</th>
<th>Source/Dest</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2023-01-05 03:32:11</td>
<td>HTTP</td>
<td>S: 10.1.5.101</td>
<td>GET - fbea0.orweb.yachts - /M/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 104.21.78.47</td>
<td></td>
</tr>
<tr>
<td>2023-01-05 03:37:20</td>
<td>ALERT</td>
<td>S: 104.21.78.47</td>
<td>ET MALWARE Likely Malicious Windows SCT Download MSXML HTTP AX M2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 10.1.5.101</td>
<td></td>
</tr>
<tr>
<td>2023-01-05 03:37:20</td>
<td>FILEINFO</td>
<td>S: 104.21.78.47</td>
<td>/ - Hostname: fbea0.orweb.yachts; Content-Type: text/html</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 10.1.5.101</td>
<td></td>
</tr>
<tr>
<td>2023-01-05 03:37:20</td>
<td>FLOW</td>
<td>S: 10.1.5.101</td>
<td>TCP 10.1.5.101:63285 -&gt; 104.21.78.47:80; Age: 20; Bytes: 5484; Packets: 13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 104.21.78.47</td>
<td></td>
</tr>
<tr>
<td>2023-01-05 03:37:20</td>
<td>FLOW</td>
<td>S: 10.1.5.101</td>
<td>TCP 10.1.5.101:63285 -&gt; 104.21.78.47:80; Age: 20; Bytes: 5484; Packets: 13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 104.21.78.47</td>
<td></td>
</tr>
</tbody>
</table>

**EveBox** - Showcasing Flow ID

[https://evebox.org/](https://evebox.org/)
What is Suricata? How it looks today?

Scirius - Showcasing Flow ID
https://github.com/StamusNetworks/SELKS
Suricata explained in one slide (IDS+NSM)

Suricata is far more than an IDS/IPS

Network Traffic
Cloud & On-premise

IDS Alerts

Protocol
Transactions

Network
Flows

PCAP
Recordings

Extracted
Files

Source: Stamus Networks
Suricata hunting - lights/rules off (NSM)

Suricata is far more than an IDS/IPS

- Alerts are only 5-10% of the data Suricata produces
- Suricata works without rules too

Source: Stamus Networks
Challenges

Adapt
Signatures evolution

- From CVE detection
  - Binary payload matching
    - Buffer overflow
    - Content triggering exploit
  - Closely bound to IPS
    - Block the payload & Protect the asset

- To .......
Signatures evolution

● To attacker behavioral analysis and infrastructure detection
  ○ Communication protocol characteristics (C2)
    ■ Type of requests (url, domain)
    ■ Client characteristics (used proto header, implementation)
  ○ Administrators behavior and process
    ■ TLS pattern in certificates, ...
● And notable events generation
  ○ Potentially interesting events: system update
  ○ Forensic usage
More protocol implementation

- Want to match on multiple protocols
  - Not a network grep anymore
- Want to log transaction on protocol
- Need complete support for more protocols
  - Application layer identification
    - Independently of the port
  - Application parsing
  - Application logging
  - Keyword to detect of the application player fields
Secure protocol implementation

- All protocols parser can suffer vulnerability
  - They parse the mud of internet
  - Protocols are complex
  - C language is not safe
    - Manual memory handling
- Big history of vulnerabilities on protocol parsers
  - Wireshark has a lot
  - Suricata has some too
Faster and safer implementation

- Use a combination
  - Rust: [https://www.rust-lang.org/](https://www.rust-lang.org/)
  - Nom: [https://docs.rs/nom/latest/nom/](https://docs.rs/nom/latest/nom/)
- Rust has rich type system and ownership mode
  - Memory safety
  - Thread safety
- Nom is parser combinator library with a focus
  - on safe parsing
  - streaming patterns
  - and as much as possible zero copy.
Rust / Nom parser example

// PORT 192,168,0,13,234,10
named!(pub ftp_active_port<u16>,
    do_parse!(
        tag!("PORT") >>
        delimited!(multispace0, digit1, multispace0) >> tag!("","") >> digit1
        >> tag!("","") >>
        digit1 >> tag!("","") >> digit1 >> tag!("","") >>
        part1: verify!(parse_u16, |&v| v <= std::u8::MAX as u16) >>
        tag!("","") >>
        part2: verify!(parse_u16, |&v| v <= std::u8::MAX as u16) >>
        (part1 * 256 + part2)
Outside evolution

- Increasing network speed
  - 40G was unthinkable
  - 100G and more is the high end now
  - More traffic means more data

- Encryption
  - Less visibility
  - No more content
  - But a lot of metadata
The Challenges

- Duplicated mirror traffic
- One side async traffic
- Cloud, on prem, Virtual infrastructure
- Needs to inspect traffic regardless of RFC specs
- Encryption
- Offloading
- Monitor this ISPs 200+Gbps link
- 2 billion logs a day+ (depending on volume/size traffic)
- OS - 64 bit/32bit/arm/Linux/Windows/BSD
The Challenges

- Duplicated mirror traffic
- One side async traffic
- Cloud, on prem, Virtual infrastructure
- Needs to inspect traffic regardless of RFC specs
- Encryption
- Offloading
- Monitor this ISPs 200+Gbps link
- 2 billion logs a day+ (depending on volume/size traffic)
- OS - 64 bit/32bit/arm/Linux/Windows/BSD
- QA anyone?
Encryption

All metadata is extracted during the clear text handshake:

- TLS SNI
- TLS Subject
- TLS Fingerprint
- TLS Issuer
- Certificate before/after dates
- JA3/JA3S
- TLS version
# What to do when the encrypted communications start:

# - default: keep tracking TLS session, check for protocol anomalies,
#    inspect tls_* keywords. Disables inspection of unmodified
#    'content' signatures.

# - bypass: stop processing this flow as much as possible. No further
#    TLS parsing and inspection. Offload flow bypass to kernel
#    or hardware if possible.

# - full: keep tracking and inspection as normal. Unmodified content
#    keyword signatures are inspected as well.

# For best performance, select 'bypass'.

#encryption-handling: default
High performance challenges

- Major perf impact factors for Suricata
  - Rules
  - Suricata version used
  - HW/OS
  - Type of traffic
Suricata - Workers mode
The RSS asymmetric hash problem

- Commodity NICs
  - Made for web/file servers to scale
  - Not build with the purpose of IDS/IPS
- IDS/IPS - needs to get both sides of a flow in the same thread, in the correct order
High performance challenges

Capture modes supported

- Netmap
- PF_RING
- AF_Packet
- AF_XDP (Suricata 7+)
- DPDK (Suricata 7+)
QAing Suricata

Many workflows and jobs

- Github
- Gitlab
- PPA Launchpad
- Suricata Verify
- Unit Tests
- Private runs

...
## QAing Suricata

### Pipeline Details

<table>
<thead>
<tr>
<th>Status</th>
<th>Pipeline Description</th>
<th>Trigger</th>
<th>Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>passed</td>
<td>Merge branch 'rebase_master6_loadtimes' into 'master'</td>
<td><img src="image" alt="Trigger Icon" /></td>
<td><img src="image" alt="Stage Icons" /></td>
</tr>
<tr>
<td>passed</td>
<td>Merge branch 'rebase_master6_loadtimes' into 'master'</td>
<td><img src="image" alt="Trigger Icon" /></td>
<td><img src="image" alt="Stage Icons" /></td>
</tr>
<tr>
<td>passed</td>
<td>Merge branch 'rebase_release_7' into 'master'</td>
<td><img src="image" alt="Trigger Icon" /></td>
<td><img src="image" alt="Stage Icons" /></td>
</tr>
<tr>
<td>passed</td>
<td>Merge branch 'rebase_release_7' into 'master'</td>
<td><img src="image" alt="Trigger Icon" /></td>
<td><img src="image" alt="Stage Icons" /></td>
</tr>
</tbody>
</table>
Sub tasks/jobs often contain thousands of checks
QAing Suricata

The final QA runs takes a few hours minimally, and generally runs overnight. It currently runs:

- extensive build tests on different OS', compilers, optimization levels, configure features
- static code analysis using cppcheck, scan-build
- runtime code analysis using valgrind, AddressSanitizer, LeakSanitizer
- ...
QAing Suricata

- regression tests for past bugs
- output validation of logging
- unix socket testing
- pcap based fuzz testing using ASAN and LSAN
- traffic replay based IDS and IPS tests
Contributing

Any feature or bug report can be publicly viewed and/or posted:
https://redmine.openinfosecfoundation.org/projects/suricata

How to contribute code:
https://suricata.io/2021/09/10/getting-started-contributing-to-suricata/

Current code PRs / reviews:
https://github.com/OISF/suricata/pulls
Conclusion

“It Has To Work.”

Global community effort

Needs to be open - roadmap, community discussions and input
More Resources

- Read the Docs: https://readthedocs.org/projects/suricata/
- More Suricata trainings/webinars: https://suricata.io/learn/
- Youtube: https://www.youtube.com/@OISFSuricata/videos
- Forums: https://forum.suricata.io/
- Awesome Suricata links: https://github.com/satta/awesome-suricata
- Discord chat: https://discord.com/invite/t3rV2x7MrG