Suricata: An Introduction

By Russ McRee – ISSA member, Puget Sound (Seattle), USA Chapter

Prerequisites

*nix operating system (Windows binaries pending)

As I prepare this month's column for your reading pleasure, I also contemplate how to pack everything I want to share with you at the ISSA International Conference (September 15-17, 2010 in Atlanta) into a 50-minute session. That challenge is not unlike how I might share everything I want you to know about Suricata in 1500 +/- words; pretty tough, but I'll give it a shot.

Suricata is the primary offering from the Open Information Security Foundation (OISF), a non-profit foundation organized to build a next generation IDS/IPS engine, with funded support from the Department of Homeland Security's Directorate for Science and Technology HOST program (Homeland Open Security Technology) and the Navy's Space and Naval Warfare Systems Command (SPAWAR).

To that end, Suricata is an open source, next generation intrusion detection and prevention engine.

It's high noon at the packet capture corral and there's a new gunslinger in town.

One of the primary missions Suricata is dedicated to is IDS/IPS that performs well in high volume network environments, and the OISF "has formed a multi-national group of the leading software developers in the security industry" to answer that call.

This project is greatly enhanced thanks to the inclusion of the likes of Ivan Ristic, one of the above mentioned leading software developers (ModSecurity) whose HTP Library (libhtp) is incorporated in Suricata. The HTP Library is an HTTP normalizer and parser that "integrates and provides very advanced processing of HTTP streams for Suricata. The HTP library is required by the engine, but may also be used independently in a range of applications and tools."[1]

This is particularly appealing when contemplating tools such as IDS/IPS in massive online services/cloud environments, the crux of my discussion at the ISSA International Conference. Arguably this talk's subtitle could be "trying to find tools that don't tip over." Suricata is meant to be one of those tools, but is subject to many of the same challenges everyone faces: load, evasion, ambiguous protocols, and faulty implementations.[2] I'll include a bit more on the HTP Library courtesy of Ivan below.

Matt Jonkman, the OISF president and board member, offered a features table that helps further exemplify Suricata's capabilities.

Features	Existing IDS/IPS Engines (Open and Commercial)	Suricata (by the OISF)
Multi-Threaded Processing	No	Yes
Complete IPv6 Support	Some	Complete
IP Reputation	Cisco Only	Yes (soon)
Automated Protocol Detection	No	Yes
GPU Acceleration	No	Yes
Multi-Platform Native Hardware Acceleration Support	No	Yes
Global Variables/Flowbits	No	Yes (soon)
Full Windows Support	Some	Yes
Inline Windows Support	No	Yes
GeoIP Lookups	No	Yes (soon)
Advanced HTTP Parsing	No	Yes
HTTP Access Logging	No	Yes
SMB Access Logging	No	Yes (soon)
HTTP Blocklist Lookups	No	Yes (soon)
Free	Some	Yes

Note the fact that multi-threaded processing is innate in Suricata but is a feature only intended for Snort 3.0, not yet available as I write this.

Consider that an unnamed military body has tested Suricata versus Snort on a large scale platform (24 processors and 128GB of RAM) and saw a very clear 6-fold speed increase over a tuned Snort implementation on the same platform.

Given all the runmode possibilities, and possibilities with heavy hitting hardware I don't have in my lab, I remind you: this is an introduction. I intend to make use of Suricata inline on some truly fat pipes, but the results will have to be in a follow-up article.

Security-aware HTTP Parsing Library

Ivan's overview clarifies some key points specific to high volume online services/cloud environments; you should contemplate these as you weigh the HTTP Parsing Library's (libhtp) features in the context of intrusion detections and web application firewalls.

1 http://blog.ivanristic.com/2009/11/http-parser-for-intrusion-detection-and-web-application-firewalls.html.

2 http://ivanr.typepad.com/files/ivan-ristic---the-challenges-of-http-intrusion-detection.pdf.

First, what is "security aware"?

HTTP's protocols and encodings are often very vague, allowing developers to interpret them in different ways going so far as to recommend flexibility for server implementations, essentially "calling" for HTTP servers to accommodate many of the client flaws. Under normal use case scenarios (typical client/server communication), the "vague" approach presents significant problems for security tools due to an "impedance mismatch" occurring when there are more than two parties involved in communication.

To mitigate this problem, one of the most important goals for the HTTP Parsing Library is to be aware of the many possible evasion tactics, to detect that evasion, flag or prevent it wherever possible, and notify the library user.[3]

In addition to security awareness the HTTP Parsing Library is designed to be secure, standards-compliant, robust, thread-safe, high-performance, flexible, modular, and extensible. Alrighty then!

As for libhtp's parsing skills?

- Request line parser • Request header parser • URI parser
- Authority parser • User-Agent header parser
- Host header parser • Response line parser
- Server response header parser • Cookie parsers
- Basic authentication parser
- Digest authentication parser
- Via request header parser
- Chunked encoding request and response body handling
- Media type and character encoding parser

Now imagine all this functionality in your open source, free IDS, with enhanced performance and well-established industry standard alerting and output.

Installing and configuring Suricata

Suricata has some dependencies to be met in advance of installation. As a Debian/Ubuntu user I'll focus on installation notes for those systems; those of you using other systems please refer to the installation guide in the doc directory on the online version.[4] This is one of those cases where reading the directions before beginning really is a good idea. I won't rehash the entirety of the installation process but will give you a few pointers I learned along the way. Many thanks to Will Metcalf, the OISF Lead QA, for his vast insight.

Most importantly, consider making use of the fact that Suricata is PF_RING capable. Luca Deri (also a member of the OISF programming team) of ntop fame offers PF_RING, a "new type of network socket that dramatically improves packet capture speed."[5] (See Figure 1[6])

There's an "INSTALL.PF_RING" doc in the Suricata-*/doc directory. While there's some work involved here, it's absolutely worth it; allow yourself the time to build Suricata with PF_RING support.

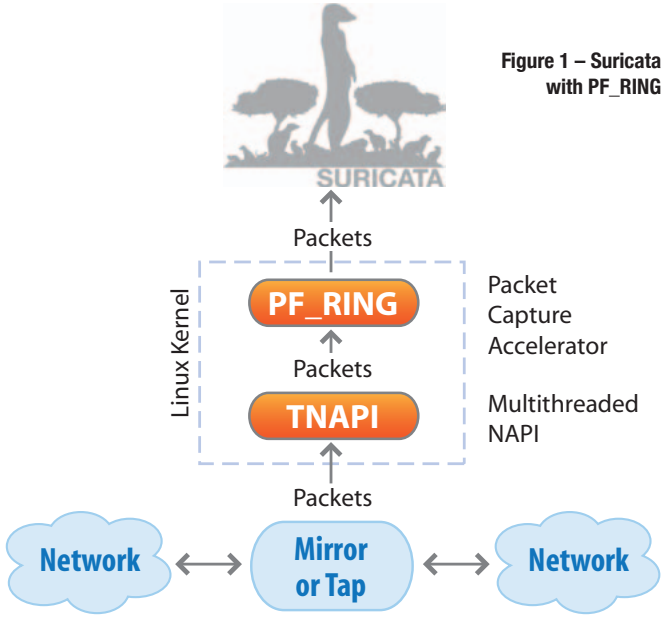

Figure 1 – Suricata with PF_RING

Assuming you've followed the detailed installation guidance, you'll execute `./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib --with-libpfring-includes=/opt/PF_RING/include --with-libpcap-libraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib" --prefix=/opt/PF_RING/` followed by `make && sudo make install`.

Running PR_RING-enabled Suricata thereafter is achieved via `/opt/PF_RING/bin/suricata --pfring-int=eth0 --pfring-cluster-id=99 --pfring-cluster-type=cluster_flow -c /etc/suricata/suricata.yaml`.

The `Suricata.yaml` file is the equivalent of `snort.conf`. You'll want to define the path to your rules, particularly if you wish to use the same set you've been using with Snort. Snort's 2.8.5.x rules are supported as well as Emerging Threats rules.[7] Note that Suricata does not support the use of `uricontent:` plus `within` as it is easily evadable and is consider by some to be bad practice. While you may see errors, the rules will load.

Victor Julien, Suricata's lead developer, also offers a tidy setup guide at Inliniac.[8]

For those of you with CUDA (Compute Unified Device Architecture) capability, have configured for CUDA support accordingly, and have loads of RAM, Will kindly devised a high-memory `suricata.yaml` file that I've posted for you.[9]

He reminded me that the Suricata threading framework is highly customizable, as are the pattern matchers, etc.; play with the `suricata.yaml` by tweaking things such as *max_pending_packets*.

3 Ivan Ristic, Security-aware HTTP parsing library overview v1.1.

4 http://www.openinfosecfoundation.org/doc/INSTALL.txt.

5 http://www.ntop.org/PF_RING.html.

6 Image modified from ntop_world, courtesy of Ntop, http://www.ntop.org/ntop_world.png.

7 http://emergingthreats.net/index.php/rules-mainmenu-38.html

8 http://www.inliniac.net/blog/2010/05/10/setting-up-suricata-0-9-0-for-initial-use-on-ubuntu-lucid-10-04.html.

9 http://holisticinfosec.org/toolsmith/files/suricata.

```
  [**] 192.168.248.118:80 -> 66.7.194.190:1112
5 07/23/10-04:47:55.960893 hosp09.land.ru [**] /del.jpg [**] Mozilla/4.0 (compatible; Synapse) [**]
  192.168.248.118:80 -> 82.204.219.223:1113
6 07/23/10-04:48:22.470228 hospnova8.pochta.ru [**] /prosen.jpg [**] Mozilla/4.0 (compatible; Synapse) [**]
  192.168.248.118:80 -> 82.204.219.221:1117
```

Figure 3 – Where'd it call home to?

I've seen mention[10] of CUDA-capable implementations showing performance upside where the run times improved as follows: 11s without CUDA, 19s with CUDA.

As an example (tweak based on your installation paths) you can test for yourself using `time` by executing `time sudo /opt/suricata/bin/suricata -c /opt/suricata/etc/suricata.yaml -r /home/<you>/your.pcap >/tmp/out.log`.

Suricata is version 1.0.0 as I write this; by the time you read this 1.0.1 or higher should be available to you.

Suricata use

With all Emerging Threats rules enabled by default (rules tuning standards still apply in production), I ran Suricata on an Ubuntu 10.04 LTS host with my Windows XP VM sandbox crunching through a variety of malware samples including Lolbot, Delfinject, and Maximus.

Expected results were written to `/var/log/suricata/fast.log` including the following alert that indicates a common malware attribute when seen propagated via URLs sent over instant messaging services:[11]

```
07/23/10-05:41:49.435766 [**] [1:2001685:7] ET MALWARE
Possible Windows executable sent when remote
host claims to send an image [**] [Classification:
A Network Trojan was detected] [Priority: 3] {6}
200.98.197.93:80 -> 192.168.248.118:1110 [Xref => http://
doc.emergingthreats.net/bin/view/Main/2001685][Xref
=> http://www.emergingthreats.net/cgi-bin/cvsweb.cgi/
sigs/MALWARE/MALWARE_Covert_Executable_DL]
```

As one might with Snort output, you can opt to write to a database and process events with the likes of Sguil, Aanval, BASE, and FPCGUI (Full Packet Capture GUI[12]). I hadn't heard of FPCGUI prior to getting myself up to speed on Suricata; I'll provide more feedback in a future article or blog post.

That said, from the oldie but goodie file, SnortSnarf will process Suricata output as easily as the above mentioned tools. This is certainly not a sustainable enterprise solution but serves to make the point that you'll have no trouble transitioning your current analysis toolset to Suricata-generated data.

Suricata produces unified2-alert output for Barnyard2 by default, along with other line-based logs and alert files. Output for the original Barnyard is available as well. Barnyard2 is a new fork under regular development. Barnyard, an output system that processes and forwards, is also a caching mechanism that is failure aware and holds alerts until the database is ready for them.[13] I highly recommend its use with Suricata.

Also enabled by default and of interest to me for quick detective work is the line-based log of HTTP requests (http.log). You won't find that in other IDS offerings (Figure 3). Cut to the quick right?

You'll find `stats.log` tracking counters by default for you as well. A snippet follows:

```
22/7/2010 -- 23:11:54
-------------------------------------------
Counter            | TM Name    | Value
-------------------------------------------
decoder.pkts       | Decode1    | 53682
decoder.bytes      | Decode1    | 23684305
decoder.ipv4       | Decode1    | 52513
decoder.ipv6       | Decode1    | 486
decoder.ethernet   | Decode1    | 53682
```

Between Suricata's multithreaded functionality, CUDA acceleration, a new and developing code base, the incorporation of Ivan's libhtp, I think you'll find ample reason to get to work with Suricata immediately.

Acknowledgements

Thanks to the following for their feedback and contributions.
—Matt Jonkman, OISF President and Board Member
—Will Metcalf, OISF Lead QA
—Victor Julien, OISF Lead Programmer
—Ivan Ristic, Programmer

In conclusion

At the onset of the article I issued the caveat that this is an introduction and I mean it, but there will be more. Suricata is immediately in the running for 2010 tool of the year and I can't wait to work with it further under more complex and demanding circumstances.

Be sure to check the OISF site; the team welcomes feedback and contributions, and is quick triage bugs if you spot one.

I'll quote Ivan as we close this month. As you contemplate your attackers remember "they need to find only one weakness in what you do; you have to protect all of them."

Suricata will certainly help you in that cause.

Cheers…until next month.

About the Author

Russ McRee, GCIH, GCFA, GPEN, CISSP, is team leader and senior security analyst for Microsoft's Online Services Security Incident Management team. As an advocate of a holistic approach to information security, Russ' website is holisticinfosec.org. Contact him at russ@holisticinfosec.org.

10 http://home.regit.org/?p=309.

11 http://holisticinfosec.blogspot.com/2010/07/messenger-abuser-malware-tactics.html.

12 http://github.com/gamelinux/fpcgui.

13 http://www.securixlive.com/barnyard2/about.php.