Global Information Assurance Certification Paper

A Performance Comparison of Intrusion Detection Systems with Regard to IPv6

GIAC (GCIA) Gold Certification

Author: Jon Mark Allen, jm@allensonthe.net

Advisor: Kees Leune

Accepted: 12 May 2015

Abstract

This paper will examine the current support of IPv6 amongst three of the most popular open source intrusion detection systems: Snort, Suricata, and Bro. It will also examine support of the IPv6 protocol within the publicly available signatures and rules for each system, where applicable. All three IDS products will be presented with the same network captures of a web application vulnerability scan - one over IPv4, and one over IPv6 - running on the same operating system installation and hardware. The performance of each system will be measured and compared first against itself and then the other two IDS products.

1 History and Overview

1.1 An Extremely Brief History of IPv6

When the Internet was first conceived, it was not expected to need to accommodate the number of devices the world now uses on a daily basis. According to a study published in August of 2014 by ABI Research, "the installed base of active wireless connected devices will exceed 16 billion in 2014, about 20% more than in 2013", and "the number of devices will more than double from the current level, with 40.9 billion forecasted for 2020" (ABI Research, 2014).

That is a problem, since IPv4 only supports around 4 billion public IP addresses. As of early 2015, there are about 7.2 billion humans in the world (U.S. Census Bureau, n.d.), though obviously not everyone has an IP address. Techniques like Network Address Translation (NAT) have also allowed us to delay moving away from IPv4, but those workarounds won't last forever.

The IETF quickly saw that the available addresses would eventually be exhausted (along with a few other issues with IPv4) and set out to develop the next version of the Internet Protocol. They apparently did not want to come up short on space with this version; a goal that they quite probably achieved when they expanded the address space from IPv4's 32 bits, supporting over 4 billion devices, to 128 bits, supporting about 3.4×10^{38} devices. That is - quite literally - an astronomical number of addresses.

As Charles Kozierok at the TCP/IP Guide website (2005) described it:

"That's about 340 trillion, *trillion*, ***trillion*** addresses. As I said, it's pretty hard to grasp just how large this number is. Consider:

- It's enough addresses for many trillions of addresses to be assigned to every human being on the planet.
- The earth is about 4.5 billion years old. If we had been assigning IPv6 addresses at a rate of 1 billion per second since the earth was formed, we would have by now used up less than one ***trillionth*** of the address space.
- The earth's surface area is about 510 trillion square meters. If a typical computer has a footprint of about a tenth of a square meter, we would have to stack

computers 10 billion high blanketing the entire surface of the earth to use up that same trillionth of the address space."

Or, as Aaron Toponce (2009) stated:

"If each IP was a single pixel, this would produce an image 18,446,744,073,709,551,616 pixels square. Now, my monitor has the capability of showing 105 pixels per linear inch. This means my monitor would need to be 2,772,778,991,358 miles in length and width if I wanted to see the image without any scrolling. Just for comparison, a light year is 5,865,696,000,000 miles. It would take almost 6 months traveling at the speed of light to start from one end of my monitor to reach the opposite."

Up to the time of this writing, the unallocated IPv4 address space is as follows (Huston, n.d.):

```
IANA Unallocated Address Pool Exhaustion:
     03-Feb-2011

Projected RIR Address Pool Exhaustion Dates:
RIR        Projected Exhaustion Date  Remaining Addresses in RIR Pool (/8s)
 APNIC:    19-Apr-2011 (actual)             0.7464
 RIPE NCC: 14-Sep-2012 (actual)             1.0245
 LACNIC:   10-Jun-2014 (actual)             0.1846
 ARIN:     15-Jun-2015                      0.2629
 AFRINIC:  23-Jan-2019                      2.6311
```

But despite the fact that the IPv4 address space is so depleted, the adoption of IPv6 has been slow in coming. However, as of early 2015, with the backing of several large telcos and hardware manufacturers, (e.g. Akamai, Google, and Cisco) at least some momentum has been gained. [1]

According to the Measurements page of the Internet Society's World IPv6 Launch website, just over 14% of the Alexa Top 1,000 websites were accessible over IPv6 (Internet Society, n.d.). What's more, since 6 June 2012 (aka "World IPv6 Launch Day"), companies like Google, Facebook, Cisco, and others have been making decent strides to push the adoption of IPv6 (Internet Society, n.d., Infographic). As of 31 January 2015, Google reports that 5.72% of their users reach Google sites over IPv6 - more than triple

[1] See http://6lab.cisco.com, http://www.akamai.com/ipv6, and https://www.google.com/ipv6.

the amount of traffic prior (ibid.). In fact, the World IPv6 Launch group estimates IPv6 will be the dominant protocol on the Internet by 2019 (Google Inc., n.d.).

1.2 Why Snort, Suricata, and Bro?

Without getting involved in the debate of which IDS is best, these three IDS solutions are certainly very common choices for both small and large organizations alike. Each product also has at least some level of support for IPv6 and is included by default in the popular Security Onion Linux distribution, which focuses on Network Security Monitoring and includes both network and host-based IDS technologies.

Snort and Suricata are considered to be signature or rule-driven IDS engines which "look at network traffic for fingerprints and identifiers that match known malicious, anomalous or otherwise suspicious traffic", While Bro aims more for an analysis driven model (Burks, n.d.). As the Security Onion documentation describes it: "Unlike rule-based systems that look for needles in the haystack of data, Bro says, 'Here's all your data and this is what I've seen. Do with it what you will and here's a framework so you can.'" (ibid.)

Each of these IDS solutions have strengths and weaknesses, and the purpose of this paper is not to judge which is "best" overall, but simply to measure and compare their adoption and support of IPv6, specifically.

1.3 Security Onion and the configuration used for the test

Security Onion is a self-contained Linux distribution, which comes with all three IDS systems pre-installed. This made it a very attractive option for this test, since the test was to be conducted on comparable hardware. All efforts were made to use out-of-the-box configurations and tuning where possible, so as not to introduce bias based on the skill of the IDS analyst. Obviously, in a production environment, both alerts and performance would be greatly improved with proper tuning to the specific environment.

Table 1, on the next page, lists the virtual machines created in VirtualBox[2] for this lab.

[2] http://www.virtualbox.org

Security Onion	eth0 (management interface)
	DHCP
	Attached to the VirtualBox NAT network for system updates.
	eth1 (monitor only interface)
	Attached to a VirtualBox Internal Network, which in this lab was named "`victimnet`").
Kali Linux	eth0
	Virtual Box Internal Network ("`victimnet`").
	Static IP assignments:
	- 10.10.2.22/24
	- fc00::2/7
Metasploitable	eth0
	- Disabled
	eth1
	VirtualBox Internal Network ("`victimnet`")
	Static IP assignments:
	- 10.10.2.21/24
	- fc00::1/7

Table 1. Virtual Machine Configurations

This test was run with the most up-to-date versions of all Security Onion components as of this writing, starting with the `securityonion-12.04.5-20140908.iso` installation image, followed by the installation of all available updates via the `soup` utility.

1.4 Description and Setup of the Traffic Captures

Each traffic capture consisted of a web application vulnerability scan performed using the OWASP Zed Attack Proxy (OWASP ZAP) – installed by default on the Kali virtual machine - against the Mutillidae web application running on the Metasploitable virtual machine, followed by a benchmark test run of "regular" traffic generated by Apache Bench, which is also installed by default on Kali.

OWASP ZAP does not support scanning IPv6 addresses directly, but has no issues scanning hostnames that resolve to an IPv6 address. Therefore, the `/etc/hosts` file on Kali was edited to include two hostnames: one resolving to the IPv4 address of

Metasploitable, and the other resolving to the IPv6 address. Using two separate names was not necessary, but did make it easier during the execution and analysis phase to keep the scans separated.

A minor configuration change was needed on the Metasploitable Apache instance, since it does not listen on IPv6 by default.

Out of the box, the `/etc/apache2/ports.conf` file looks like this:

```
Listen 0.0.0.0:80

<IfModule mod_ssl.c>
    Listen 0.0.0.0:443
</IfModule>
```

Once the IPv6 address was applied to the interface, the `ports.conf` file was updated as follows to listen on both IPv4 and IPv6 addresses:

```
Listen 10.10.2.21:80
Listen [fc00::1]:80

<IfModule mod_ssl.c>
    Listen 10.10.2.21:443
    Listen [fc00::1]443
</IfModule>
```

If you're following along at home, please note that you will need to be root or utilize `sudo` to be able to edit the `ports.conf` file. The Apache services were then restarted (`/etc/init.d/apache2 restart`) -- again, as root -- to apply the changes.

Traffic for both the scan and the "regular" requests was captured by running tcpdump on the Kali VM. For ease of measurement, separate capture files were used for the OWASP ZAP scan and the Apache Bench traffic, e.g.:

```
root@kali:~# tcpdump -i eth1 -w ipv4_owasp_zap.pcap -nv
```

The default scanning profile from OWASP ZAP was used for all scans. Once the scan was complete, the tcpdump capture was stopped, and a new capture started:

```
root@kali:~# tcpdump -i eth1 -w ipv4_ab_4000_25.pcap -nv
```

For comparison, from a different terminal, Apache Bench was then used to generate a reasonable number of "normal" HTTP requests:

```
root@kali:~# ab -n 4000 -c 25 \
http://metasploitable.localhost/mutillidae/
```

Again for the sake of clean packet captures, when the scans and traffic generation were completed for IPv4, the tcpdump capture was stopped, and a new one started to capture the IPv6 traffic. The OWASP ZAP scan was then repeated, but with the IPv6 hostname, instead. This was followed by yet a fourth tcpdump capture and a new Apache Bench run. (Note that with sufficient resources, either on one machine or multiple, both attack traffic and "legitimate" requests could be executed simultaneously for a somewhat more accurate representation of IDS performance.)

Before these packet capture files were ready to be replayed across the monitoring interface of Security Onion, the checksums had to be recalculated[3]. While this could be accomplished via the tcprewrite utility, for this paper, a small python script[4] was created, using the Scapy library ("Scapy," n.d.).

2 Security Onion IDS Configuration

Security Onion is configured via the custom `sosetup` GUI utility, located on the Desktop. This configuration includes the selection of Snort vs Suricata, as well as several other pertinent settings. The "Advanced Setup" mode was used to more finely control the lab environment.

This instance was setup as a "standalone" server, so that it served as both the alert sensor as well as the database and analysis server. For both Snort and Suricata, the IDS rule set utilized the GPL Emerging Threats feed, in addition to the registered (free) VRT rule set, which is available from snort.org.

Since the philosophies and features of Bro compared to the other two IDS solutions are so different, the setup script always asks if Bro should be enabled, regardless of which previous IDS engine was selected.

Along with network IDS functionality, Bro is capable of extracting executable files found crossing the network, and also sending the HTTP query logs Bro generates to

[3] Snort and Suricata can be configured to ignore the checksums of replayed traffic, but recalculation was preferred here to more closely simulate real-world traffic.

[4] The author is by no means a seasoned Python wrangler, so the script is *truly* awful, but it was effective for the purposes of this test. ☺

ELSA or Sguil. While these are excellent features and truly help Security Onion to shine as a network analysis platform, they were not necessary for our testing and so were disabled in this environment.

Argus, PRADS, full packet capture, and Salt were each disabled via the `sosetup` script for all tests. In addition, Security Onion utilizes OSSEC, by default, as a host-based IDS. While OSSEC is an excellent tool, it was also disabled after startup via the init scripts[5]:

```
sudo /etc/init.d/ossec-hids-server stop
```

Before updating the IDS environment variables, the sensor processes were stopped:

```
SO-user@so-sensor:~$ sudo nsm_sensor_ps-stop
```

Then the `/etc/nsm/[sensor_name_int]/snort.conf` file was modified, as seen below:

```
SO-user@so-sensor:/etc/nsm/so-sensor-eth1$ diff -u1 snort.conf
snort-updated.conf
--- snort.conf 2015-03-23 20:26:34.000000000 +0000
+++ snort-updated.conf    2015-05-01 01:27:57.011717468 +0000
@@ -44,3 +44,3 @@
 # Setup the network addresses you are protecting
-ipvar HOME_NET [192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]
+ipvar HOME_NET [10.10.2.0/24,fc00::/7]

@@ -295,3 +295,3 @@
    min_response_seconds 5
-preprocessor stream5_tcp: policy windows, detect_anomalies,
require_3whs 180, \
+preprocessor stream5_tcp: policy linux, detect_anomalies,
require_3whs 180, \
    overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
```

These changes updated the `HOME_NET` variable to include the specific internal networks used in this lab and configured the `stream5` pre-processor reassembly policy to match the OS of the web server being defended.

The Suricata engine allows for different stream reassembly policies for different networks, and by default was already configured for Linux hosts on the network segment used in this lab. Therefore, only the `HOME_NET` variable required updating in the `suricata.yaml` file (located in the same directory as the `snort.conf` file):

[5] Though this decision was revisited later in the evaluation.

```
SO-user@so-sensor:~$ diff -u1 suricata.yaml suricata-updated.yaml
--- suricata.yaml    2015-04-30 21:07:56.000000000 -0500
+++ suricata-updated.yaml 2015-04-30 21:06:59.000000000 -0500

@@ -943,3 +943,3 @@

-    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
+    HOME_NET: "[10.10.2.0/24,fc00::/7]"
```

When these changes had been completed, the IDS engine processes were restarted:

```
SO-user@so-sensor:~$ sudo nsm_sensor_ps-restart
```

At this point, the network captures were replayed across the `eth1` interface of Security Onion:

```
SO-user@so-sensor:~$ sudo tcpreplay -t -i eth1 \
ipv4_owasp_zap_fixed.pcap && \
sudo tcpreplay -t -i eth1 ipv4_ab_fixed.pcap
```

The `-t` option to tcpreplay stands for "top speed" and replays the packets back across the interface as fast as possible. Once all alerts were logged and documented, the IPv6 traffic was replayed:

```
SO-user@so-sensor:~$ sudo tcpreplay -t -i eth1 \
ipv6_owasp_zap_fixed.pcap && \
sudo tcpreplay -t -i eth1 ipv6_ab_fixed.pcap
```

With the data from Snort generated and documented, the Security Onion configuration files were modified to use the Suricata IDS and each of the packet captures re-replayed across the IDS `eth1` interface to generate the data for the Suricata evaluation. The modification of the configuration files was as simple as the following commands[6]:

```
sudo nsm_sensor_ps-stop
sudo sed -i 's|ENGINE=snort|ENGINE=suricata|g' \
/etc/nsm/securityonion.conf
sudo rule-update
sudo nsm_sensor_ps-start
```

[6] https://github.com/Security-Onion-Solutions/security-onion/wiki/FAQ#im-currently-running-snort--how-do-i-switch-to-suricata

3 Snort IDS

3.1 Description of IPv6 Support in Snort

The earliest mention of IPv6 in the Snort change log shows IPv6 support beginning to appear in February 2000, with the release of version 1.6 occurring the following month (Snort, 2014).

Some quick grep commands reveal that, as of this writing, there were 64 IPv6 specific rules in the VRT+ET rule set, with 29 actually enabled by default in Security Onion:

```
SO-user@so-sensor:/etc/nsm/rules$ grep -i ipv6 \
downloaded.rules | grep -c alert
64

SO-user@so-sensor:/etc/nsm/rules$ grep -i ipv6 \
downloaded.rules | grep -cv '#'
29
```

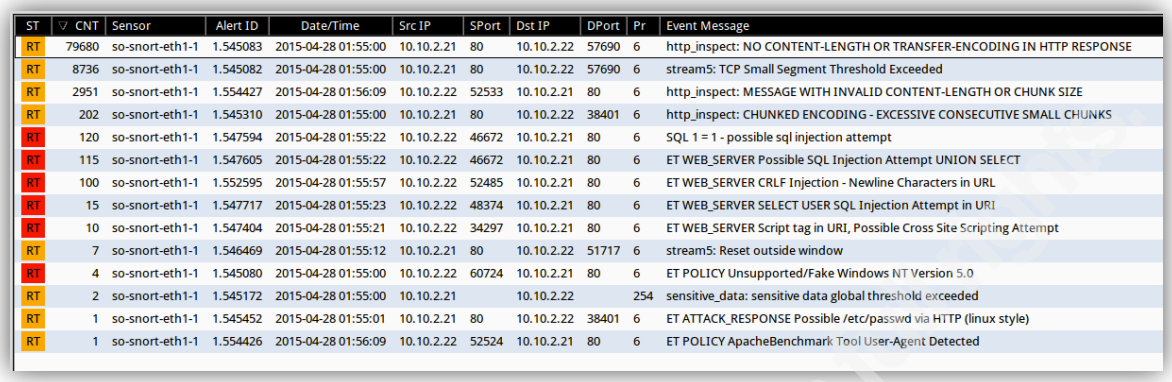
Table 2 shows the breakdown of the classes of IPv6-specific signatures currently available.

Type of Signature	Count
ICMP6 protocol alerts	24
IPv6 protocol decode messages	24
Metasploit meterpreter binding	8
Other	8

Table 2. Types of IPv6 signatures in the VRT+ET rule sets

Signatures have historically been driven by attack discovery and disclosure, for obvious reasons, so it is likely these counts will go up, and the variety of classes increase as IPv6-specific attacks are discovered and become more prevalent.

Figure 2 shows the alerts generated against the IPv4 traffic. The IPv4 traffic triggered a total of 91,994 alerts, of which 91,578 were from the Snort pre-processors, and the remaining 366 were from the shared VRT+ET rule sets. Of the pre-processor generated alerts, the vast majority (79,680) were generated by the `http_inspect` pre-processor, with nearly all the rest coming from the `stream5` pre-processor.

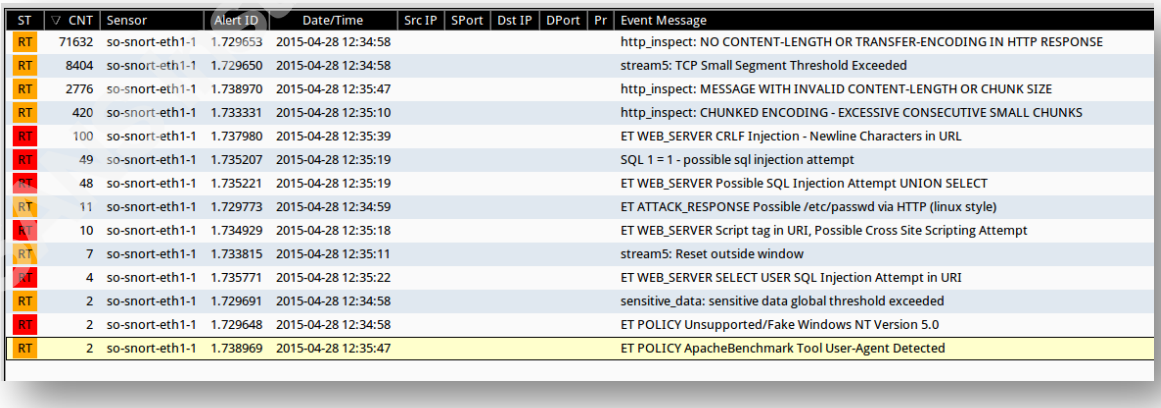


ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	79680	so-snort-eth1-1	1.545083	2015-04-28 01:55:00	10.10.2.21	80	10.10.2.22	57690	6	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
RT	8736	so-snort-eth1-1	1.545082	2015-04-28 01:55:00	10.10.2.21	80	10.10.2.22	57690	6	stream5: TCP Small Segment Threshold Exceeded
RT	2951	so-snort-eth1-1	1.554427	2015-04-28 01:56:09	10.10.2.22	52533	10.10.2.21	80	6	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE
RT	202	so-snort-eth1-1	1.545310	2015-04-28 01:55:00	10.10.2.21	80	10.10.2.22	38401	6	http_inspect: CHUNKED ENCODING - EXCESSIVE CONSECUTIVE SMALL CHUNKS
RT	120	so-snort-eth1-1	1.547594	2015-04-28 01:55:22	10.10.2.22	46672	10.10.2.21	80	6	SQL 1 = 1 - possible sql injection attempt
RT	115	so-snort-eth1-1	1.547605	2015-04-28 01:55:22	10.10.2.22	46672	10.10.2.21	80	6	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
RT	100	so-snort-eth1-1	1.552595	2015-04-28 01:55:57	10.10.2.22	52485	10.10.2.21	80	6	ET WEB_SERVER CRLF Injection - Newline Characters in URL
RT	15	so-snort-eth1-1	1.547717	2015-04-28 01:55:23	10.10.2.22	48374	10.10.2.21	80	6	ET WEB_SERVER SELECT USER SQL Injection Attempt in URI
RT	10	so-snort-eth1-1	1.547404	2015-04-28 01:55:21	10.10.2.22	34297	10.10.2.21	80	6	ET WEB_SERVER Script tag in URI, Possible Cross Site Scripting Attempt
RT	7	so-snort-eth1-1	1.546469	2015-04-28 01:55:12	10.10.2.21	80	10.10.2.22	51717	6	stream5: Reset outside window
RT	4	so-snort-eth1-1	1.545080	2015-04-28 01:55:00	10.10.2.22	60724	10.10.2.21	80	6	ET POLICY Unsupported/Fake Windows NT Version 5.0
RT	2	so-snort-eth1-1	1.545172	2015-04-28 01:55:00	10.10.2.21		10.10.2.22		254	sensitive_data: sensitive data global threshold exceeded
RT	1	so-snort-eth1-1	1.545452	2015-04-28 01:55:01	10.10.2.21	80	10.10.2.22	38401	6	ET ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)
RT	1	so-snort-eth1-1	1.554426	2015-04-28 01:56:09	10.10.2.22	52524	10.10.2.21	80	6	ET POLICY ApacheBenchmark Tool User-Agent Detected

Figure 1. Snort alerts against IPv4 traffic, as shown in Sguil.

Of the signature-based rules, the classes of alerts generated were what would be expected: SQL injection, cross site scripting, CRLF injection, and data exfiltration (i.e. retrieval of the `/etc/password` file).

The IPv6 traffic triggered a total of 83,467 alerts, shown in Figure 3 below, of which 83,241 were from the Snort pre-processors, and the remaining 226 were from the shared VRT+ET rules. The ratio of pre-processor to signature-based alerts was roughly the same as for the IPv4 traffic, as 71,632 of the alerts were from the same rule as above. Additionally, the signature-based alerts were similar to what was seen in the IPv4 scan.



ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	71632	so-snort-eth1-1	1.729653	2015-04-28 12:34:58						http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
RT	8404	so-snort-eth1-1	1.729650	2015-04-28 12:34:58						stream5: TCP Small Segment Threshold Exceeded
RT	2776	so-snort-eth1-1	1.738970	2015-04-28 12:35:47						http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE
RT	420	so-snort-eth1-1	1.733331	2015-04-28 12:35:10						http_inspect: CHUNKED ENCODING - EXCESSIVE CONSECUTIVE SMALL CHUNKS
RT	100	so-snort-eth1-1	1.737980	2015-04-28 12:35:39						ET WEB_SERVER CRLF Injection - Newline Characters in URL
RT	49	so-snort-eth1-1	1.735207	2015-04-28 12:35:19						SQL 1 = 1 - possible sql injection attempt
RT	48	so-snort-eth1-1	1.735221	2015-04-28 12:35:19						ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
RT	11	so-snort-eth1-1	1.729773	2015-04-28 12:34:59						ET ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)
RT	10	so-snort-eth1-1	1.734929	2015-04-28 12:35:18						ET WEB_SERVER Script tag in URI, Possible Cross Site Scripting Attempt
RT	7	so-snort-eth1-1	1.733815	2015-04-28 12:35:11						stream5: Reset outside window
RT	4	so-snort-eth1-1	1.735771	2015-04-28 12:35:22						ET WEB_SERVER SELECT USER SQL Injection Attempt in URI
RT	2	so-snort-eth1-1	1.729691	2015-04-28 12:34:58						sensitive_data: sensitive data global threshold exceeded
RT	2	so-snort-eth1-1	1.729648	2015-04-28 12:34:58						ET POLICY Unsupported/Fake Windows NT Version 5.0
RT	2	so-snort-eth1-1	1.738969	2015-04-28 12:35:47						ET POLICY ApacheBenchmark Tool User-Agent Detected

Figure 2. Snort alerts against IPv6 traffic, as shown in Sguil.

But what stood out most was the fact that Sguil did not display the IPv6 addresses at all. In fact, the ability to drill down into the packet, or show any other details of the alert, did not work when viewing an alert from an IPv6 stack. While troubleshooting this issue,

it was discovered that barnyard2, the dedicated log parser for the unified2 log format, does not yet support IPv6 addresses ("Snort mailing list archives," 2013). Since all other IDS management interfaces in Security Onion feed off this parser, none of them are able to show this information.

4 Suricata IDS

4.1 Description of IPv6 Support in Suricata

Suricata uses the same rule set as Snort, so it had the same 64 IPv6 rules (29 enabled by default) as Snort did. But it also has an additional `decoder-events.rules` file with an additional 32 decoder rules, 31 of which were enabled by default in Security Onion, for a total of 96 rules, with 60 enabled by default.

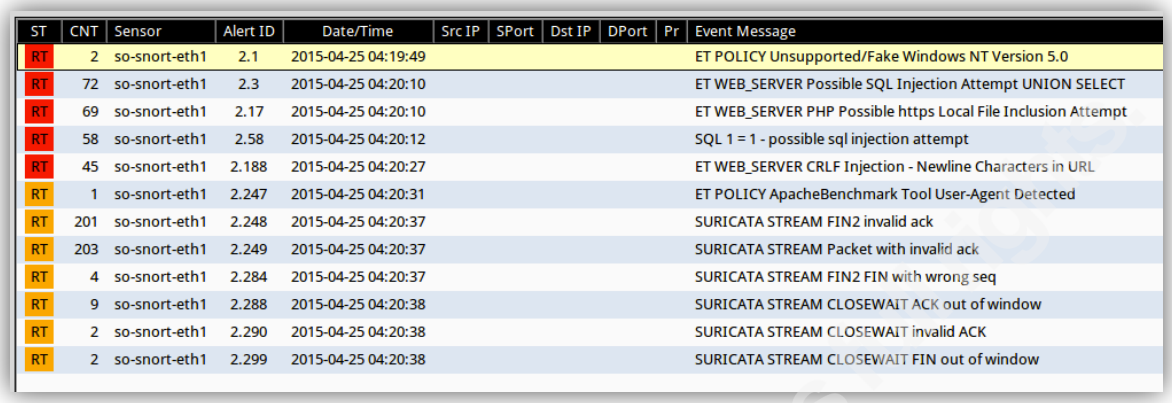
The IPv4 traffic triggered a total of 286 alerts, shown in Figure 4, of which 9 were from the Suricata `STREAM` signatures, and the remaining 277 were from the shared VRT+ET rule sets.

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	2	so-snort-eth1	2.953	2015-04-25 04:32:56	10.10.2.22	52532	10.10.2.21	80	6	ET POLICY ApacheBenchmark Tool User-Agent Detected
RT	2	so-snort-eth1	2.669	2015-04-25 04:32:03	10.10.2.22	60724	10.10.2.21	80	6	ET POLICY Unsupported/Fake Windows NT Version 5.0
RT	38	so-snort-eth1	2.898	2015-04-25 04:32:45	10.10.2.22	52485	10.10.2.21	80	6	ET WEB_SERVER CRLF Injection - Newline Characters in URL
RT	77	so-snort-eth1	2.673	2015-04-25 04:32:09	10.10.2.22	49785	10.10.2.21	80	6	ET WEB_SERVER PHP Possible https Local File Inclusion Attempt
RT	95	so-snort-eth1	2.696	2015-04-25 04:32:16	10.10.2.22	46672	10.10.2.21	80	6	ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
RT	10	so-snort-eth1	2.683	2015-04-25 04:32:14	10.10.2.22	34297	10.10.2.21	80	6	ET WEB_SERVER Script tag in URI, Possible Cross Site Scripting Attempt
RT	53	so-snort-eth1	2.711	2015-04-25 04:32:17	10.10.2.22	49029	10.10.2.21	80	6	SQL 1 = 1 - possible sql injection attempt
RT	2	so-snort-eth1	2.951	2015-04-25 04:32:55						SURICATA ICMPv6 unknown type
RT	7	so-snort-eth1	2.671	2015-04-25 04:32:07	10.10.2.21	80	10.10.2.22	56964	6	SURICATA STREAM ESTABLISHED retransmission packet before last ack

Figure 3. Suricata alerts against IPv4 traffic, as shown in Sguil.

The classes of signature-based alerts generated were similar to that of Snort, with the addition of local file inclusion, but lacked the data exfiltration detection.

As seen in Figure 5, the IPv6 traffic triggered a total of 668 alerts, of which 421 alerts were from the Suricata `STREAM` signatures, and the remaining 247 were from the shared VRT+ET rule sets.



ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	2	so-snort-eth1	2.1	2015-04-25 04:19:49						ET POLICY Unsupported/Fake Windows NT Version 5.0
RT	72	so-snort-eth1	2.3	2015-04-25 04:20:10						ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT
RT	69	so-snort-eth1	2.17	2015-04-25 04:20:10						ET WEB_SERVER PHP Possible https Local File Inclusion Attempt
RT	58	so-snort-eth1	2.58	2015-04-25 04:20:12						SQL 1 = 1 - possible sql injection attempt
RT	45	so-snort-eth1	2.188	2015-04-25 04:20:27						ET WEB_SERVER CRLF Injection - Newline Characters in URL
RT	1	so-snort-eth1	2.247	2015-04-25 04:20:31						ET POLICY ApacheBenchmark Tool User-Agent Detected
RT	201	so-snort-eth1	2.248	2015-04-25 04:20:37						SURICATA STREAM FIN2 invalid ack
RT	203	so-snort-eth1	2.249	2015-04-25 04:20:37						SURICATA STREAM Packet with invalid ack
RT	4	so-snort-eth1	2.284	2015-04-25 04:20:37						SURICATA STREAM FIN2 FIN with wrong seq
RT	9	so-snort-eth1	2.288	2015-04-25 04:20:38						SURICATA STREAM CLOSEWAIT ACK out of window
RT	2	so-snort-eth1	2.290	2015-04-25 04:20:38						SURICATA STREAM CLOSEWAIT invalid ACK
RT	2	so-snort-eth1	2.299	2015-04-25 04:20:38						SURICATA STREAM CLOSEWAIT FIN out of window

Figure 4. Suricata alerts against IPv6 traffic, as show in Sguil.

The Suricata `STREAM` processor alerts were significantly increased against the IPv6 traffic, but there was no change in the classes of alerts seen from the signature-based rules. Again, since Suricata depends on barnyard2 for log parsing into the Sguil DB, the IPv6 addresses and other related details were not available for this traffic.

5 Bro IDS

5.1 Description of IPv6 Support in Bro

The earliest mention of IPv6 in the Bro change log shows support beginning to appear in April 2012, with version 0.17-8 ("Detailed Version History — Bro 2.3.2 documentation," n.d.). In testing for this paper, Bro had no problems decoding and displaying IPv6 traffic and alerts. But while the author did not find any IPv6 specific rules, it bears repeating that the philosophy of Bro is significantly different than either of the other two IDS solutions under examination.

As stated previously, while Snort and Suricata are pattern-based engines, Bro focuses more on traffic pattern analysis, and as such, the resulting logs and alerts are significantly different. It should also be noted, that Sguil/Squert do not currently support ingesting logs from Bro, but that also means that reviewing alerts was not impacted by the barnyard2

deficiency. Analysis of the Bro alerts during this test was fully conducted from the terminal.[7]

After executing the OWASP ZAP scans, Bro entered the following in the `notice.log` file, as viewed with the `bro-cut` utility[8]:

```
SO-user@so-sensor:~$ bro-cut note msg src \
 < /nsm/bro/logs/current/notice.log
HTTP::SQL_Injection_Attacker    An SQL injection attacker was
discovered!    10.10.2.22
HTTP::SQL_Injection_VictimAn SQL injection victim was discovered!
   10.10.2.21
HTTP::SQL_Injection_Attacker    An SQL injection attacker was
discovered!    fc00::2
HTTP::SQL_Injection_VictimAn SQL injection victim was discovered!
   fc00::1
```

Bro did not alert on each instance of a SQL injection attempt, but rather reported that an attacker had been detected. This would make monitoring alerts significantly easier than parsing through literally thousands of individual alerts. However, in this lab, Bro did not alert on any other classes of attacks, such as cross-site scripting or command injection.

Additionally, each web request seen is recorded in the `http_INTERFACE.log` file:

```
SO-user@so-sensor:~$ bro-cut id.orig_h id.resp_h method uri \
 < /nsm/bro/logs/current/http_eth1.log
fc00::2 fc00::1 POST
/mutillidae/index.php?page=login.php'+AND+'1'='1
fc00::2 fc00::1 POST
/mutillidae/index.php?page=login.php+AND+1=1
fc00::2 fc00::1 POST
/mutillidae/index.php?page=login.php+AND+1=1+--+
10.10.2.22      10.10.2.21      GET
/mutillidae/index.php?page=redirectandlog.php"+AND+"1"="1"+--+
10.10.2.22      10.10.2.21      GET
/mutillidae/index.php?page=redirectandlog.php+AND+1=1
10.10.2.22      10.10.2.21      POST
/mutillidae/index.php?page=login.php"+AND+"1"="1
```

Reviewing other Bro logs did not reveal significantly additional information regarding this particular scan.

[7] ELSA could also have been used for analysis of each of these systems, though that would not support the drill-down features even for IPv4 traffic.

[8] https://www.bro.org/sphinx/components/bro-aux/README.html#bro-cut

6 Overall Comparisons and Workarounds

Snort performed roughly the same for both IPv4 and IPv6 traffic, with some reduction in signature-based alerts on IPv6 traffic. Snort also produced prolific alerts based primarily on the stream5 pre-processor reassembly process, but an IDS administrator should be able to tune those alerts down to more reasonable levels.

Compared to Snort, Suricata's performance between IPv4 and IPv6 was more consistent, at least in regard to the signature-based alerts. But the stream reassembly process in Suricata did not handle the IPv6 traffic well, relative to IPv4 traffic.

Bro generated the exact same number of alerts on both IPv4 and IPv6, but in this case, it only detected one class of web attack traffic, leaving the remainder of attacks for detection by some other means.

Management of alerts from both Snort and Suricata are significantly hamstrung by the lack of support in barnyard2 for logging of IPv6 addresses. As seen in the screenshots included above, none of the graphical management interfaces installed by default support the display or storage of IPv6 addresses, due to the lack of support in barnyard2. The current version of barnyard2 is 2.1-13, and according to the barnyard2-users email list archives, IPv6 support is planned for 2.2 ("Snort mailing list archives," 2013). Unfortunately, the last stable version at the time of this writing was released on 14 May 2013 – nearly two full years ago.

6.1 Alternate Logging Options

One possible resolution to this problem would be to completely bypass barnyard2 for log collection, and instead use fast or full format alert files to allow OSSEC or some other external syslog server or event correlation utility to do alert correlation and management. However, this approach is not without drawbacks, either.

To enable the full logging format for Snort, a small change must be made to the `snort.conf` file, after first stopping the `nsm_sensor_ps` processes:

```
SO-user@so-sensor:/etc/nsm/so-sensor-eth1$ diff -u1 snort.conf
snort-updated.conf
--- snort.conf 2015-03-23 20:26:34.000000000 +0000
+++ snort-updated.conf    2015-05-01 01:27:57.011717468 +0000
@@ -533,5 +533,6 @@
 # output unified2: filename merged.log, limit 128, nostamp,
mpls_event_types, vlan_event_types
```

```
-output unified2: filename snort.unified2, limit 128
+#output unified2: filename snort.unified2, limit 128

 # Additional configuration for specific types of installs
+output alert_full: snort-full.log
 # output alert_unified2: filename snort.alert, limit 128, nostamp
```

This simply disables all pre-existing output formats and adds the `alert_full` output plugin[9].

The process for Suricata is very similar, though it does not currently support the `alert_full` format, and so is limited to `alert_fast`. Again, changes are made to the `suricata.yaml` file, located in the same directory as the `snort.conf` file:

```
SO-user@so-sensor:~$ diff -u1 suricata.yaml suricata-updated.yaml
--- suricata.yaml    2015-04-30 21:07:56.000000000 -0500
+++ suricata-updated.yaml 2015-04-30 21:06:59.000000000 -0500
@@ -81,3 +81,3 @@
   - fast:
-      enabled: no
+      enabled: yes
       filename: suricata-fast.log

@@ -114,3 +114,3 @@
   - unified2-alert:
-      enabled: yes
+      enabled: no
       filename: snort.unified2
```

When these changes have been completed, restart the IDS engine processes:

```
sudo nsm_sensor_ps-restart --only-snort-alert
```

The log file is then located in the `/nsm/sensor_data/[sensor_name_int]/` directory.

OSSEC can be modified to monitor these alert files by adding additional `<localfile>` directive(s) to the `/var/ossec/etc/ossec.conf` file:

```
<localfile>
  <log_format>snort-full</log_format>
  <location>/nsm/sensor_data/[sensor_name_int]/snort-
full.log</location>
</localfile>

<localfile>
   <log_format>snort-fast</log_format>
```

[9] Snort could be configured to log in the alert_fast format instead, but that option is omitted here to avoid redundancy.

```
   <location>/nsm/sensor_data/[sensor_name_int]/suricata-
fast.log</location>
</localfile>
```

Please note that root access is required for all OSSEC modifications. After restarting OSSEC (`sudo service ossec-hids-server restart`), it will begin monitoring the alert logs generated by Snort/Suricata. Without any additional configuration, IDS alerts in OSSEC are recorded as follows (for `alert_fast` formatted logs) in the `alerts.log` file:

```
** Alert 1430968061.51807: - securityonion,syslog,
2015 May 07 03:07:41 so-snort->/nsm/sensor_data/so-snort-
eth1/suricata-fast.log
Rule: 20101 (level 4) -> 'IDS event.'
05/07/2015-03:07:37.875905  [**] [1:2006446:11] ET WEB_SERVER
Possible SQL Injection Attempt UNION SELECT [**] [Classification:
Web Application Attack] [Priority: 1] {TCP}
fc00:0000:0000:0000:0000:0000:0000:0002:33725 ->
fc00:0000:0000:0000:0000:0000:0000:0001:80
```

Unfortunately, OSSEC parsing of the `alert_full` format seems to be less consistent:

```
** Alert 1430969114.7017254: - securityonion,syslog,
2015 May 07 03:25:14 so-snort->/nsm/sensor_data/so-snort-
eth1/snort-1/snort-full.log
Rule: 20101 (level 4) -> 'IDS event.'
[**] [1:30041:2] SQL 1 = 1 - possible sql injection attempt [**]

** Alert 1430969114.7018006: mail  - syslog,errors,
2015 May 07 03:25:14 so-snort->/nsm/sensor_data/so-snort-
eth1/snort-1/snort-full.log
Rule: 1002 (level 2) -> 'Unknown problem somewhere in the system.'
[Classification: Potentially Bad Traffic] [Priority: 2]

** Alert 1430969114.7018268: - securityonion,syslog,
2015 May 07 03:25:14 so-snort->/nsm/sensor_data/so-snort-
eth1/snort-1/snort-full.log
Rule: 20101 (level 4) -> 'IDS event.'
[**] [129:12:1] Consecutive TCP small segments exceeding threshold
[**]
```

Ideally, the OSSEC log parsers would be updated to parse and display - at a minimum - the source and destination IP address information, and additional correlation rules should be created for maximum benefit. While outside the scope of this paper, the example modifications above give a general idea of the process that would need to be followed.

Jon Mark Allen, jm@allensonthe.net

One other possibility for exploration is the use of an ELK stack (Elasticsearch, Logstash, and Kibana) for management of the alerts generated; however there was neither time nor space in this report to fully explore that option.

7 Conclusion

IPv6 traffic adoption is rising, and while the IDS engines evaluated support the new protocol, there is still much work to be done before they are ready to meet the demands of everyday consumption. There are very few IPv6-specific signatures developed and released to the public so far, but that is potentially due to the lack of IPv6-specific attacks discovered and disclosed, to date. Bro IDS shows parity both in detection and alert management of attacks over IPv6, but given the different detection strategy, it should be layered with a more traditional IDS engine for a more complete picture of the classes of attacks in progress. For both Snort and Suricata, the actual detection of attacks over IPv6 is comparable to similar attacks conducted over its predecessor, but the ability to manage those alerts is woefully, and surprisingly, lacking. But this finding should certainly not be taken as an indictment of Security Onion as a whole. One available option for alert management within Security Onion could include the use of OSSEC, but even in that case, additional work is needed to produce a useable solution.

Jon Mark Allen, jm@allensonthe.net

References

ABI Research. (2014, August 20). The Internet of Things will drive wireless connected devices to 40.9 billion in 2020 | ABI Research. Retrieved April 5, 2015, from https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect

Burks, D. (n.d.). Introduction to Security Onion · Security-Onion-solutions/security-onion wiki · GitHub. Retrieved April 5, 2015, from https://github.com/Security-Onion-Solutions/security-onion/wiki/IntroductionToSecurityOnion

Detailed Version History — Bro 2.3.2 documentation. (n.d.). Retrieved April 19, 2015, from https://www.bro.org/sphinx/install/changes.html

Google Inc. (n.d.). IPv6 – Google. Retrieved February 3, 2015, from http://www.google.com/intl/en/ipv6/statistics.html

Huston, G. (n.d.). IPv4 address report. Retrieved April 5, 2015, from http://www.potaroo.net/tools/ipv4/

Internet Society. (n.d.). Infographic | World IPv6 Launch. Retrieved February 3, 2015, from http://www.worldipv6launch.org/infographic/

Internet Society. (n.d.). Measurements | World IPv6 Launch. Retrieved February 3, 2015, from http://www.worldipv6launch.org/measurements/

Kozierok, C. M. (2005, September 20). The TCP/IP Guide - IPv6 address size and address space. Retrieved April 5, 2015, from http://www.tcpipguide.com/free/t_IPv6AddressSizeandAddressSpace-2.htm

Scapy. (n.d.). Retrieved April 20, 2015, from http://www.secdev.org/projects/scapy/

Snort mailing list archives. (2013, April 27). Retrieved April 30, 2015, from http://seclists.org/snort/2013/q2/416

Snort. (2014, December 24). changelog_2972.txt. Retrieved April 19, 2015, from https://www.snort.org/downloads/snort/changelog_2.9.7.2.txt

Toponce, A. (2009, March 8). The sheer size of IPv6. Retrieved April 5, 2015, from https://pthree.org/2009/03/08/the-sheer-size-of-ipv6/

U.S. Census Bureau. (n.d.). Population Clock. Retrieved April 5, 2015, from http://www.census.gov/popclock/

Jon Mark Allen, jm@allensonthe.net