5-2022

Comparative Study of Snort 3 and Suricata Intrusion Detection Systems

Cole Hoover
University of Arkansas, Fayetteville

Citation
Hoover, C. (2022). Comparative Study of Snort 3 and Suricata Intrusion Detection Systems. *Computer Science and Computer Engineering Undergraduate Honors Theses* Retrieved from https://scholarworks.uark.edu/csceuht/105

Comparative Study of Snort 3 and Suricata Intrusion Detection Systems

An Undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering
College of Engineering
University of Arkansas
Fayetteville, AR
April, 2022

by

Cole Hoover

Comparative Study of Snort 3 and Suricata Intrusion Detection Systems

Cole Hoover
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
cdhoover@uark.edu

Dale R. Thompson
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
drt@uark.edu

Abstract— **Network Intrusion Detection Systems (NIDS) are one layer of defense that can be used to protect a network from cyber-attacks. They monitor a network for any malicious activity and send alerts if suspicious traffic is detected. Two of the most common open-source NIDS are Snort and Suricata. Snort was first released in 1999 and became the industry standard. The one major drawback of Snort has been its single-threaded architecture. Because of this, Suricata was released in 2009 and uses a multithreaded architecture. Snort released Snort 3 last year with major improvements from earlier versions, including implementing a new multithreaded architecture like Suricata. This paper compares Suricata and the new and improved Snort 3 based on their performance and alert behavior. Both NIDS were installed on the same system, configured with the default recommended configurations, used default rulesets, and evaluated the same malicious traffic. In this analysis, both NIDS performed very similar in their resource utilization, but when analyzing the malicious traffic, Suricata detected more attacks than Snort 3 using their standard rulesets.**

Keywords—NIDS, Snort, Suricata, performance, rules, comparison

I. Introduction

Intrusion detection systems are a piece of software that sit on the network and monitor the traffic for anything that is deemed malicious. There are many different variations of intrusion detection systems. True intrusion detection systems only monitor the traffic and send alerts if any traffic is found to be illegitimate. An IDS can be configured to block or suspend traffic that triggers an alert, and these are called intrusion prevention systems. Within IDSs, there are network intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). Host-based IDSs monitor a device or devices on the network while network intrusion detection systems monitor traffic on a whole network. There are also two types of IDSs: anomaly-based and signature-based. Anomaly-based IDSs compare traffic to a baseline of trusted traffic and alert if new traffic deviates from the baseline. Signature-based IDSs compare traffic to a set of signatures or rules that define suspicious activity on a network and alert if any pattern of traffic matches a signature. In this paper, we specifically deal with signature-based network intrusion detection systems, namely Snort and Suricata. When NIDS are placed at strategic points on the network, they can monitor all traffic that comes in and out of the network for any malicious activity. This traffic is analyzed and compared to the list of rules, and when a piece of traffic matches an attack pattern, it is logged, and an alert is sent.

Rules are one of the most important aspects of a signature-based IDSs. The rules define what traffic is valid and what is invalid. They are what the traffic is compared to when it is analyzed for suspicious activity. Rules also specify the action to be taken once any unusual activity is detected, such as alert, log, or pass. There are several organizations that put out their own rulesets created by security researchers and community members who analyze attacks and what traffic they generate, and then create rules that detect this traffic. Custom rules can also be created by users to detect specific things in their own environment. Rules are meant to be combined, interchanged, and tuned to allow for the best detection in each specific environment and to reduce the number of false positives.

There are several metrics that can be used to evaluate intrusion detection systems. Antonatos et al. offers attack detection rate, false positives, and capacity to test performance [1]. Research from Mell suggests coverage, probability of false alarm, probability of detection, attack resistance, ability to handle high bandwidth traffic, and capacity to test accuracy [2]. In work from Albin [3], they evaluated performance by measuring CPU utilization, memory use, and network use. In this paper, the metrics that will be used are CPU and memory utilization, as well as the number of alerts generated during an attack.

Snort is one of the most popular and widely used open-source NIDS. It was originally created by Martin Roesch in 1999 as a lightweight intrusion detection system [4] and has grown to a very powerful IDS with many features. It can be used as a packet sniffer, a packet logger, or a complete IDS or IPS. Snort was developed using a single threaded architecture and has stayed that way despite the trends in multicore processors, until recently. The Snort team released Snort 3 in 2021 which implements a multithreaded architecture among several other upgrades and improvements. According to Snort, it has enhanced performance, faster processing, and improved scalability [5]. Snort 3 was rewritten in C++, making it more modular, introduces threading and shared memory to allow for scaling and multiple packet processing, offers more than 200 plugins for more customization, and the rule syntax has been updated for easier comprehension and is more concise to decrease the amount of rules and allow them to run faster [5].

Suricata is another leading open-source NIDS released by the Open Information Security Foundation in 2010. It was primarily started in response to the growing use of multicores in computers and to make up for what Snort was missing. It shares many features with Snort and boasts extra features including automatic protocol detection, file extraction, logging HTTP requests, DNS queries, and TLS certificates, and has a multithreaded engine [6]. It implements Lua scripting to detect things rules cannot as well as advanced outputs to allow for easy integration with many logging tools.

This paper details a comparison of Snort 3 and Suricata IDSs, observing their alerting behavior when examining the same malicious traffic and using the rulesets specifically designed for their systems, as well as examining their performance when analyzing captured traffic stored in a pcap file. The next section discusses related work to comparing Snort and Suricata. Section III looks at the architectures of Snort 3 and Suricata. Section IV compares the rules and rule syntax that Snort and Suricata use. The evaluation of the IDSs is done in Section V and the conclusions are presented in Section VI.

II. Related Work

There have been numerous studies conducted that compare Snort and Suricata against a large array of metrics. Most of the work has been done with Snort 2, as Snort 3 was only released last year, although a few recent studies have been done evaluating Snort 3. Most of the research is done comparing the two NIDS in specific environments, such as high-speed networks, limited resource environments, or environments with several cores and excess resources to examine their performance, looking at resource utilization and packet processing speeds. Other studies compare the accuracy of detecting attacks, false positive rates, and other metrics that deal with the differences in the rulesets provided for Snort and Suricata. Some papers combine both the performance and detection aspects. This paper is similar in that regard, examining the alerting behavior of the IDSs with their respective rulesets, while at the same time measuring their performance and resource utilization, as well as continuing the necessary research on Snort 3, as it is a newer product.

One of the earlier works that looks at both Snort and Suricata is “A Comparative Analysis of the Snort and Suricata Intrusion Detection Systems” by Eugene Albin, published in 2011 [3]. Suricata was released the previous year and was still in its early stages. He ran three experiments to measure performance and accuracy in a busy network environment. The first experiment tested the live performance of both NIDS, measuring resource utilization. The second experiment tested Suricata on a supercomputer to measure processing speed. The last experiment tested the accuracy of both NIDS when evaluating malicious traffic. The experiments showed that Suricata was more resource intensive than Snort, had the ability to be configured and scaled for boosted performance with multiple cores, and that tuning of rulesets to avoid false positives and false negatives is necessary. It was concluded that Snort was still a strong and capable NIDS, but Suricata could handle higher volumes of traffic and as network throughput continued to increase, Suricata could scale to match it. Work done by Day and Burns in 2011 [7] is similar Albin’s, but tested the accuracy of both NIDS under stress, measured capacity, coverage, false positive and false negative rates. The results indicated that Suricata requires more overhead than Snort but when multiple cores are available, it is more accurate. Research conducted by White et al. in 2013 [8] expands on the work done by Day and Burns and scaled the resources available to the NIDS as well as varied the configurations, rulesets, and workloads. The results showed that Suricata outperformed Snort throughout all tests and used less resources than Snort.

More recent works include “Evaluating Network Intrusion Detection Systems for High-Speed Networks” by Hu et al. in 2017 [9], “Comparing the Performance of Intrusion Detection Systems: Snort and Suricata” by Brandon Murphy in 2019 [10], and “Dynamical analysis of diversity in rule-based open source network intrusion detection systems” by Asad and Gashi in 2021 [11]. Hu et al. examined Snort, Suricata, and Bro, another popular open-source IDS. They investigate resource usage, packet processing speeds, and packet drop rate of the NIDS when in a high-speed network and compare performance with different configurations and traffic volumes. They conclude Suricata performs better because of its multithreaded architecture, but Snort and Bro could be configured to achieve improved performance. In the dissertation by Brandon Murphy [10], a thorough study is done to measure the accuracy, efficiency, and resiliency of Snort and Suricata. Suricata was shown to be more accurate, but Snort used less memory, and there was no statistically significant difference in the dropped packets by Snort and Suricata. Lastly, in the work done by Asad and Gashi [11], Snort and Suricata were compared by their configurational and functional diversity. To compare the configurational diversity, the rulesets and Blacklisted IP Address lists were investigated. To compare the functional diversity, the alerting behavior of the NIDS was studied when analyzing network traffic. It was concluded that there is significant diversity in both the rulesets and Blacklisted IP Address lists which led to significant differences in alerting behavior.

III. IDS Architectures

Snort was originally built to best suit the computers and networks of its time. It started as a lightweight packet sniffer, running on single core computers, and handled the much smaller network traffic when it was released. It had a single threaded startup and a single packet thread per process. It used up more memory, using memory to store duplicate information for each process. Now that multi-core computers are the new norm and extensive research has been done to show Suricata’s superior performance due to its multithreaded architecture, Snort decided to rewrite its engine in C++ from C to support multithreading as well as a host of new features and improved old features that led to better and faster performance. It has a multithreaded startup for faster initialization, supports multiple packet threads, reduces memory usage by sharing information with each process, allowing for more memory for packets, supports more than 200 plugins, boosts the data acquisition and pcap readback speeds, allows for more configuration of certain features, has a more simplified default configuration requiring less tuning, and enhances rule writing [5].

The old architecture was more linear and not as optimized as the new Snort 3 [5]. First, traffic is acquired from a live network or pcap file. The packets are then sent through packet decoder routines that identify the packet structure for link level protocols as well as ports. Packets are then sent through a set of preprocessors. Each preprocessor checks to see what kind of packets it has collected and what their behavior is. Then, the packets are sent through the detection engine, where it checks each packet against the rules and is then logged and dealt with based on what action the rule specified. Snort 3's packet processing was redesigned to be more flexible using an event driven approach [5], and a diagram of this is shown in Fig. 1. It was optimized to produce data when needed instead of every time to reduce normalizations, which presents the packet data in a standard format. Instead of using preprocessors and iterating over a list to test each one, it uses what they call inspectors. Inspectors create inspector events that can supply data to other inspectors. This data is not published, just the access to the data so that the data only needs to be normalized once on the first access, resulting in just-in-time processing. These inspection events along with a variety of plug-in inspectors (represented by the other in the diagram) allow for a much more flexible and efficient packet processing system.

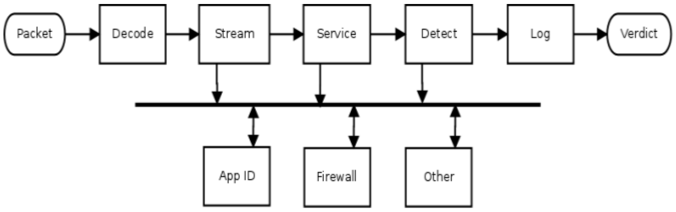

Fig. 1. Snort 3's Packet Processing Architecture

Suricata was partly created in response to the need for a multithreaded IDS. It borrows a lot of functionality from Snort, and is similar in several aspects, but also implemented some functionality Snort was missing. It can function as a NIDS, NIPS, network security monitor, and pcap logger [6]. It has a scalable flow engine, a TCP stream engine, and an IP Defrag engine. It can parse protocol at the link layer and application layer and has a stateful HTTP parser that logs transactions and can extract files. Its detection engine is very powerful, highly configurable, and able to be tuned to exact needs.

One of the most important features is its multithreading capabilities. It has fully configurable threading and is able to run anywhere from a single thread to dozens. It also has optional CPU affinity settings where Suricata designates threads to a set CPU or across however many CPUs the system has. Suricata has four thread-modules for processing packets [12] which is shown in Fig. 2. First is the packet acquisition module that collects packets from the network or from a pcap file. Next is the decode and stream application layer module. It decodes the packets based on their protocols and ports, performs stream-tracking where it checks that a correct network connection is made, reconstructs the original stream of packets, and finally inspects the application layer. Then, the detection module, which can have several detection threads running simultaneously, compares the traffic to the rules. Finally, action is taken based on what the rules specified, and events will be logged. The process is very similar to Snort but has its own unique qualities.

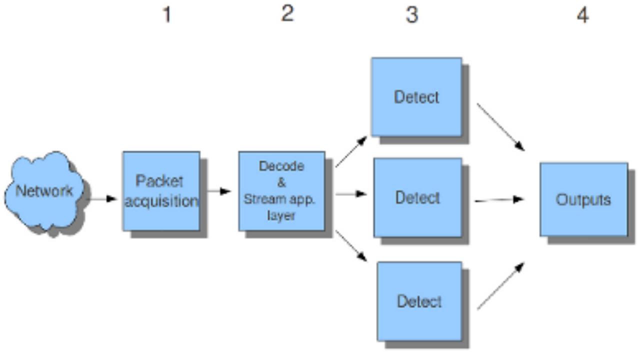

Fig. 2. Suricata's Multithreaded Detection Architecture

IV. IDS Rules

Rules are an integral part for signature-based IDSs to function. They add the advantage of zero-day detection, unlike signatures, which are written after an attack has already happened. When writing rules, the focus is on detecting a vulnerability, not an exploit or a unique piece of data [5]. The syntax of these rules, the number of rules, and the way they are parsed can affect an IDS's performance. Snort 3 has improved rule parsing as well as similar rule syntax to Snort 2, but with improvements to make them easier to write and understand. Suricata's rule syntax is the same as Snorts but offers different functionality. Both will be discussed below.

Snort 2 rules have a rule header and rule options [5]. The header is made up of the action to be taken, the protocol or type of traffic, such as TCP, UDP, ICMP, or IP, the source IP address, the source port, a direction operator indicating which way the traffic is flowing, the destination IP address and the destination port. After the header is the rule options, which add power and flexibility to rules. There are general rule options as well as detection options. General rule options include message, to include what the rule is detecting, flow, to help filter rules that only apply to certain directions of traffic, reference, to include references for the rule, class type, which tells what the effect of the detected attack is, and signature ID, which helps to identify rules. The detection options are content, which tells rules what to look for in the packet payload, PCRE, which allows rules to be written in PCRE to aid in more complex matching, and byte test to check for a number of bytes. While this syntax worked well, it was somewhat inconsistent and was able to be improved in Snort 3 [5]. In the new syntax, the elements protocol, networks, ports, and direction operators are now optional, allowing for shorter and less redundant rules, whereas before, these elements were required. A new protocol keyword http was added in Snort 3 which allows for better HTTP detection, whereas in Snort 2, HTTP detection was defined in the content option and required more detail. Snort 3 also adds new sticky and dynamic buffer selectors which helps reduce redundancy in rules. Also, the file keyword was added as an option after the rule action which helps reduce the number of rules written to detect the same traffic over different protocols and directions. Lastly, rule metadata is now true metadata in Snort 3, not affecting detection.

Snort also improved its use of shared object (SO) rules. These are written in the Shared Object rule language, which is similar to C, and must be compiled in order to use them. These rules allow for detection not possible with the Snort rule language. They can be configured to detect more conditions than regular rules. These rules also allow for the obfuscation of exact detection in the normal rule language [5].

Suricata rules also have an action that determines what to do when a rule matches, a rule header that defines the protocol, IP addresses, ports, and direction of traffic, and rule options that define more specific things in the rule. Suricata adds a few extra action keywords, as well as several application layer or layer seven protocol key words that Snort does not support [12]. There are several other nuances in configuration that differ between Snort and Suricata in terms of rules and rule writing but they are technical and beyond the scope of this discussion. Suricata was written in a way that the Snort rulesets were supported but required some tuning to fix some of the rules that use different keywords and other minor differences. Now that Snort 3 has a newer rule syntax, Snort 3 rules are not compatible in Suricata.

V. Evaluation

This section details all aspects of the experimental setup and design. The goal of this experiment was to evaluate the performance of two multithreaded intrusion detection systems and their alerting behavior when analyzing the same malicious traffic against their respective rulesets. To measure performance, CPU utilization and memory utilization statistics were collected, and to investigate the alerting behavior, extra logging was configured for each IDS.

A. Experimental Setup

The environment used was a virtual machine running Ubuntu 18.04.6 LTS as the operating system. The virtual machine used 4 cores and had 4 GB of RAM and 50 GB of disk space. Both Suricata and Snort were installed on the virtual machine and were tested individually. Suricata version 6.0.4 was used and is the latest stable release. Snort version 3.1.6.0 was used for the experiment, but smaller updates took place during the course of the experiment, and the latest version is 3.1.21.0. In order to maintain continuity of the experiment, Snort was not updated to the most recent release. PulledPork is the rule management tool used for Snort. It downloads the latest rule files from Snort's website. When Snort 3 was released, PulledPork was also rewritten from Perl to Python 3 and is called PulledPork3. Both tools were installed on the system and tested, but PulledPork3 is ultimately what was used as the rule management tool because of its improved features and better compatibility with Snort 3. Version 3.0.0.4 of PulledPork3 was used. Suricata's rule management tool is Suricata-Update [13]. Starting from version 4 of Suricata, it comes bundled with Suricata, so it did not have to be installed separately. It was used to manage the Suricata rulesets. Command line tools htop and nmon were used to measure CPU and memory utilization.

Two different rulesets were used in the experiment, Snort's Talos LightSPD registered ruleset and Emerging Threats ET Open ruleset for Suricata. Talos is a group of network security experts who work to maintain and write rules for Snort. Snort offers 3 different rulesets: the community ruleset, a registered user ruleset, and a subscription ruleset. The community ruleset is provided by the vast community of Snort users, and they are free to use by anyone. It is updated daily and a subset of the subscription ruleset. The registered ruleset is 30 days behind the subscriber ruleset. You must register an account with Snort to use this ruleset. The subscriber ruleset is the most up to date and comprehensive ruleset and is paid for by a subscription. This ruleset is developed, tested, and approved by the Talos team, and is specifically written for Snort. Emerging Threats is a division of Proofpoint and put out the Emerging Threats Rulesets [14]. They offer the ET Open and ET Pro rulesets. The ET Open ruleset is free and maintained by members of the security community. The ET Pro ruleset is maintained by the Emerging Threats research team. These rules are generally written to work best with Suricata but can also be used in Snort 2. They are not supported in Snort 3 as of now. A breakdown of the rulesets and number of rules used in testing is shown in Table I.

TABLE I. Ruleset Details

Rulesets	Number of Rules
Talos LightSPD	20,359
ET Open	24,956

The malicious traffic that is analyzed in this experiment comes from the work of White et al. [8]. They posted the pcap files of captured runs of Pytbull traffic used in their experiment. Pytbull is an open-source framework used to test IDSs. There are two different traces used in this experiment. One is of a single denial of service attack, and the other is a combined trace of all 8 tests that were run in Pytbull. The eight tests were client-side download attacks, basic ruleset testing, non-RFC compliant packets, fragmented payloads, multiple failed logins, evasion techniques, shell codes, and the before mentioned DoS attack. The pcap files were read directly by the NIDS during testing.

In this experiment, both Snort and Suricata were installed on the same virtual machine running Ubuntu using the recommended base configurations. To facilitate collecting alert behavior, some extra logging was enabled on both systems. Using PulledPork3 and Suricata-Update, the Snort and Suricata rulesets were updated to the most current version. Then, the command line tools nmon and htop were started to observe the resource utilization for each run. Snort was then run, passing it the configuration file and the first pcap file of the DoS attack. The resource utilization data and the number of alerts generated was collected and recorded, and the log files were examined to investigate what was alerted on. Snort was reset and the data collection tools were reset as well. Then Snort was run again, using the same command, but passing it the larger pcap file of the 8 Pytbull tests. The data was again collected and recorded and the log files were investigated. Next, Suricata was run, passing it the configuration file and the first pcap file of the DoS attack. The resource utilization data and alerts generated was collected and recorded, and the log files were studied. The tools were reset and Suricata was run on the second trace of the 8 Pytbull tests. The data was collected and log files inspected. This concluded the experiment.

B. Results

The following section presents the results of resource utilization and alert behavior from the testing of Snort and Suricata and an analysis of these results.

Suricata and Snort performed very similar when tested in terms of their resource utilization. Both used roughly 25% of the CPU throughout all tests. Each NIDS used only one core at 100%, resulting in an average CPU utilization of 25% because the system has 4 cores. This was an interesting result as each IDS supports multithreading and was configured to use this functionality and should have distributed more among the other cores. Suricata showed to be slightly more memory intensive. In Snort's first test on the smaller pcap file of the DoS attack, it used 4% of RAM and in the second test it used 5%. In previous studies of Snort 2, Snort's memory use was also lower than Suricata, showing Snort 3 maintains the small overhead it takes in packet processing, even enhancing it in the new architecture. Suricata used approximately 7.5% of RAM in both tests, which is interesting considering the second pcap trace with 8 tests is significantly larger than the first trace with only the DoS attack. Suricata's use of more memory is consistent with findings in previous research, showing that Suricata's multithreaded architecture is less efficient in memory utilization than Snort 3. Fig. 3 shows the memory and CPU usage from the first test and Fig. 4 shows the results for the second test.

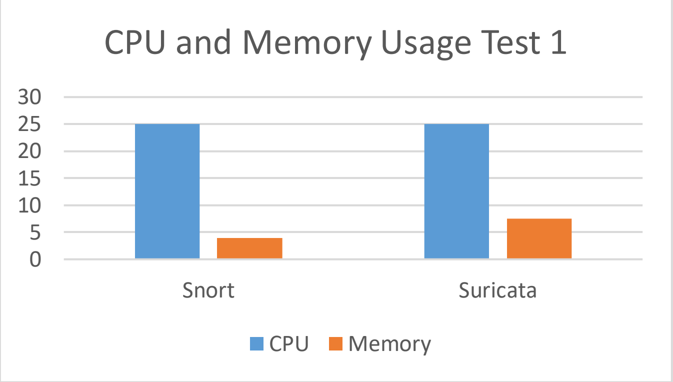

Fig. 3. Snort and Suricata CPU and memory usage by percentage in Test 1

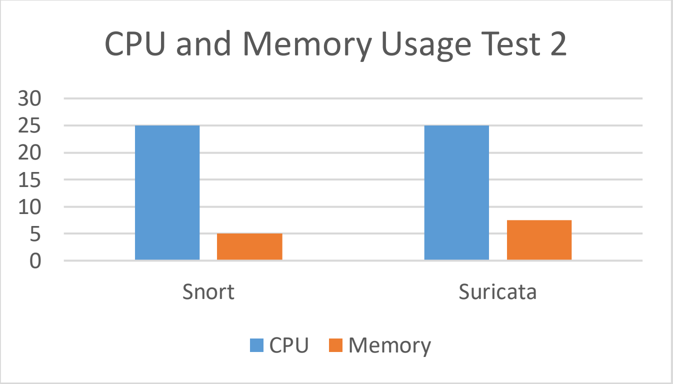

Fig. 4. Snort and Suricata CPU and memory usage by percentage in Test 2

The two NIDS performed very differently in their alerting behavior. Suricata alerted nearly 2 times more when analyzing the first trace and 3.5 times more when analyzing the second trace. When analyzing the first trace, Snort alerted 1,211 times to Suricata's 2,281 times. For the second trace, Snort alerted 2,944 times and Suricata alerted 10,441 times. Fig. 5 and Fig. 6 show the number of alerts for Snort and Suricata for both tests. Suricata had almost 4,600 more rules enabled than Snort when inspecting the traffic which could have led to it alerting and detecting more suspicious traffic. Another possible reason for this difference is because the rulesets put out by Talos and Emerging Threats have many differences between the two. In a study done by Asad and Gashi [11], Snort and Suricata rules were compared based on the content rule option field, and only 1% of rules resulted in a match. Lastly, the ET Open ruleset, which was used in Suricata, is mostly created from members of the security community and the Talos LightSPD ruleset used in Snort 3 is created and maintained by dedicated Talos researchers, which could be a cause of the large discrepancy in rules and alerting behavior.

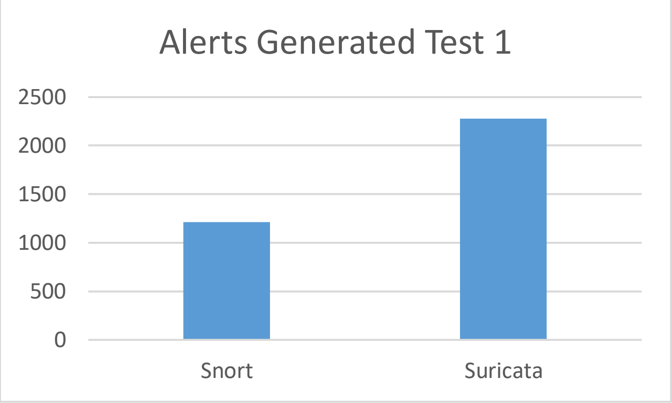

Fig. 5. Snort and Suricata alerts in Test 1

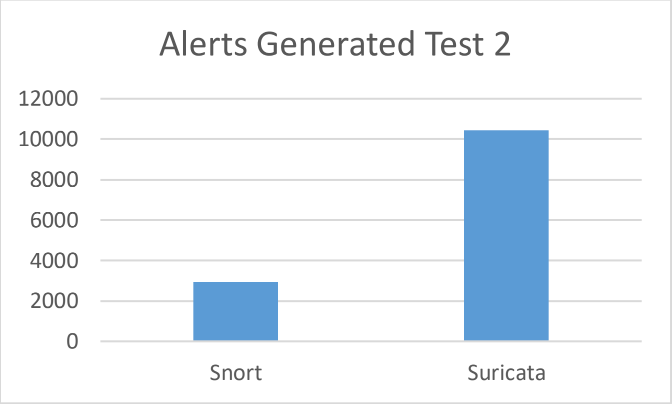

Fig. 6. Snort and Suricata alerts in Test 2

C. Issues

A few issues were encountered during the experiment process. First, there was a problem when trying to install Suricata onto the system. The installation guide from the Suricata documentation was followed, but when installing the packages, an error was received when it was configuring libhyperscan4 that said the CPU lacked support for the Supplemental Streaming SIMD Extensions 3 (SSSE3) instruction set that was required to execute programs linked against hyperscan. An option was given to install the package and when no was selected, the installation completed, but Suricata would not start. Suricata was then deleted, and different commands were used to install the packages, and this time yes was selected to go ahead and install the package. It completed the install and started working as normal. Also, the initial goal of this paper was to compare the NIDS against the same rulesets, similar to the previous work for a more accurate comparison. When it was discovered that Snort 3 had changed the rule syntax, the ET Open ruleset did not support Snort 3, and that the Snort 3 rulesets did not work in Suricata, the goal was shifted. Instead of measuring just performance, it was decided to also examine the alerting behavior of both NIDS with their respective rulesets specifically tailored for each system.

VI. Conclusion

In this paper, Suricata and Snort 3, two popular open-source network intrusion detection systems were compared based on their performance and alerting behavior. Snort has been the industry standard, despite only being single threaded and performing weaker in modern environments compared to Suricata, which is multithreaded and highly scalable. Snort 3 is a complete rewrite of the old engine to support multithreading and several new features to compete with Suricata. Both multithreaded NIDS are tested on the same system and inspect the same malicious traffic. The NIDS are configured using the default recommended settings and use the most recent rulesets designed specifically for each system. The results show that the systems are very similar in CPU utilization, showing Snort 3's multithreading is up to par, but in memory usage, Suricata is more memory intensive than Snort. Regarding alerting behavior, Suricata detected more attacks and alerted 2 times and 3.5 times more than Snort 3 in our tests. The rules were not tuned, and false positive rates were not measured, but this exemplifies the vast differences in the ET Open and Talos LightSPD rulesets that are designed for Suricata and Snort respectively as well as the differences in the detection engines of the two NIDS. Previous work has been done comparing Snort 2 and Suricata, but as Snort 3 was publicly released in early 2021, there have been few studies researching it, and more work is needed to fully examine the new NIDS with its improved features and better performance.

References

[1] S. Antonatos, K. Anagnostakis, and E. Markatos, "Generating Realistic Workloads for Network Intrusion Detection Systems," in *4th ACM Workshop on Software and Performance*, Jan. 2004, vol. 29, no. 1, pp. 207–215, Accessed: Mar. 14, 2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/974044.974078.

[2] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, "An Overview of Issues in Testing Intrusion Detection Systems," National Institute of Standards and Techonology, Gaithersburg, MD, Jul. 2003. Accessed: Mar. 14, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7007.pdf.

[3] E. Albin, "A Comparative Analysis of the Snort and Suricata Intrusion-Detection Systems," Master's Thesis, Naval Postgraduate School, 2011.

[4] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of LISA '99: 13th Systems Administration Conference*, Seattle, WA, Nov. 1999, pp. 229–238, Accessed: Nov. 16, 2022. [Online]. Available: https://www.usenix.org/legacy/event/lisa99/full_papers/roesch/roesch.pdf.

[5] Snort, "Snort - Network Intrusion Detection & Prevention System," *Snort.org*, 2019. https://www.snort.org/.

[6] "Home," *Suricata*. https://suricata.io/.

[7] D. Day and B. Burns, "A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines," in *ICDS 2011*, Gosier, Guadeloupe, France, Feb. 2011, pp. 187–192, Accessed: Oct. 21, 2021. [Online].

[8] J. White, T. Fitzsimmons, and J. Matthews, "Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata," in *Cyber Sensing 2013*, Baltimore, MD, May 2013, vol. 8757, Accessed: Oct. 13, 2021. [Online]. Available: https://people.clarkson.edu/~jmatthew/publications/SPIE_SnortSuricata_2013.pdf.

[9] Q. Hu, M. R. Asghar and N. Brownlee, "Evaluating network intrusion detection systems for high-speed networks," 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1-6, doi: 10.1109/ATNAC.2017.8215374

[10] B. Murphy, "Comparing the Performance of Intrusion Detection Systems: Snort and Suricata," Doctoral Dissertation, Colorado Technical University, 2019.

[11] H. Asad and I. Gashi, "Dynamical analysis of diversity in rule-based open source network intrusion detection systems," *Empirical Software Engineering*, vol. 27, no. 4, Oct. 2021, doi: 10.1007/s10664-021-10046-w.

[12] "Suricata User Guide — Suricata 6.0.4 documentation," *suricata.readthedocs.io*. https://suricata.readthedocs.io/en/suricata-6.0.4/index.html (accessed Nov. 01, 2021).

[13] "suricata-update - A Suricata Rule Update Tool — suricata-update 1.3.0dev0 documentation," *suricata-update.readthedocs.io*. https://suricata-update.readthedocs.io/en/latest/ (accessed Nov. 01, 2021).

[14] P. Schroeder, "Emerging Threats FAQ," *Emergingthreats.net*, Nov. 11, 2018. https://doc.emergingthreats.net/bin/view/Main/EmergingFAQ (accessed Mar. 10, 2022).