An Empirical Evaluation of Suricata for Protecting Vulnerable Web Application Servers

Zachary M. Etters
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
[redacted]

Alexander L. Borzillo
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
[redacted]

Krittanat Kulsakdinun
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
[redacted]

Abstract

This research project evaluates Suricata as an Intrusion Detection and Prevention System (IDPS) by deploying it in a virtualized lab environment. This will enable the research study to simulate real-world network cyberattacks and mitigations. To achieve this goal, this research study implemented a pfSense firewall, a Damn Vulnerable Web Application (DVWA) server, and a Kali Linux machine all within a virtualized environment. These virtual machines will enable the research study to generate realistic network traffic that can be used to test Suricata's functionality. This allows the research project to examine Suircata's ability in detecting network port scans, Denial-of-Service (DoS) attempts, and SQL injection exploits. This research study's findings showed that Suricata reliably detected network scans and DoS attacks by default. However, SQL injection attacks required additional configuration and a custom ruleset for alerting. These findings highlight both the strengths and weaknesses of Suricata as an IDS and demonstrate the need for continuous ruleset maintenance to ensure that Suricata functions correctly. The findings also show that additional layers of protection are needed within production network environments to ensure the safety of critical infrastructure.

1 Introduction

As the number of data breaches and cyberattacks increases each year, cybersecurity professionals must be aware of potential tools that are available to them that will help prevent potential cybersecurity attacks. These tools are extensively used in the information technology industry, and consequently, it is important to gain practical experience using them. This research paper explores a specific category of tools and implements them in a practical testing environment that emulates a real-world network environment. By conducting research in this approach, we will gain practical experience using these

tools and be able to identify some of their strengths and weaknesses. Applying these tools in a real-world environment will showcase the importance of intrusion detection systems in preventing potential cybersecurity attacks.

One of the ways to prevent these potential cybersecurity attacks from occurring within your network is by using intrusion detection. Intrusion detection is defined as a "process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices" (Scarfone & Mell, 2007, p. 15). Scarfone and Mell (2007) further describe that these incidents are caused by "malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized" (p. 15). However, not all incidents that are logged and monitored are malicious in nature (Scarfone & Mell, 2007). It is important that a cybersecurity professional can distinguish between these false positives and identify the real threats. This process can be made easier by using an Intrusion Detection System (IDS).

2 Methodology

This research project evaluates Suricata as an Intrusion Detection and Prevention System (IDPS) for protecting web application servers from common attack vectors. To achieve this, the study uses a virtualized lab environment to simulate realistic network conditions and attack vectors. The purpose of this methodology is to test Suricata's detection capability with real-world attacks against web application servers. However, in order for the study to be conducted, we must first build the lab environment.

2.1 Lab Environment

To build out the lab environment, we are using VMware Workstation Pro 17 v17.6.4. This will allow us to utilize a virtual, recreatable network environment to evaluate Suricata's abilities. First, we created a host-only virtual network, VMNet 12, within VMware Workstation. We configure that virtual network by turning off the DHCP server and by giving that network the private IP address of `192.168.99.0/24`. By configuring these options, we gain complete control over how network devices are addressed within that network. See Figure 1.

Our next step was to install a pfSense virtual machine. We did so by grabbing the ISO online and creating a virtual machine within VMware Workstation Pro. We changed the virtual machine type to FreeBSD 10 and ensured that the VM had enough resources. We assigned the VM two CPU cores, two gigabytes of RAM, and a 20-gigabyte hard disk. The VM was then assigned three network adapters. The first network adapter is for the external-facing side or the Wide Area Network (WAN). This adapter was configured to be NATed to the host's network. The second network adapter is for the internal-facing side or the Local Area Network (LAN) and is set to VMNet 12, which is the newly created host-only network. The last network adapter is for Suricata so that it can monitor the network traffic, and it is also set to VMNet 12. After the virtual machine was created and configured, we then installed pfSense from the ISO.

The pfSense installation was left to defaults for the most part, aside from creating a user account to log into. After pfSense was installed from the ISO, it rebooted, and we began configuring the network interfaces. For the WAN interface, we let it grab the IP address and the default gateway from DHCP from the host. For the LAN side, we assigned it the address of `192.168.99.2`. This will be the new default gateway for all devices placed within VMNet 12. For the last adapter, we did not assign any addresses, as Suricata will use this adapter in promiscuous mode to monitor the network traffic. After the network configuration was completed, we were able to access pfSense's Web GUI from the host machine.

After entering the credentials we configured during the installation of pfSense, we immediately updated pfSense to version 2.8.1. Once the update was complete, we took a snapshot of the virtual machine to serve as a backup in case we would need to revert the virtual machine at any time. We then enabled the DHCP server on pfSense to handle allocating IP addresses to any devices that we would connect to VMNet 12. Once that was completed, we could then begin creating the web application server virtual machine. See Figure 2.

2.1.1 Web Application Server

To install a web application server in our virtualized lab environment, we first need to know how we would plan on exploiting it. Rather than designing our own platform to test attacks on, we chose a solution that requires less time to build. We opted to install Damn Vulnerable Web Application (DVWA). DVWA is a purpose-built web application server that is designed to be vulnerable to many different exploits (Wood, 2025). DVWA has a web interface that we can access using the IP address of the virtual machine on which it is installed. DVWA contains many different attack vectors that can be exploited in our lab environment (Wood, 2025). Some of the more prominent ones are SQL Injection, Cross-Site Scripting (XSS), and brute-forcing the admin panel (Wood, 2025). This web application will allow us to test many different exploits in the hopes that Suricata will detect the attacks being executed.

Installing DVWA was rather simple in practice. We first created an Ubuntu virtual machine within our VMware Workstation Environment. We gave it 4 CPU cores, 8 GB of RAM, and a 64 GB hard disk. This gives the virtual machine more than enough resources to run DVWA. Next, we ran through the GUI installation of Ubuntu, creating a user account and the hard-disk partition. After Ubuntu was installed, we restarted the virtual machine and ensured that it had network connectivity. The virtual machine grabbed an IP address of `192.168.99.100` from the pfSense DHCP server. Next, we updated the packages to their latest version using `sudo apt update && sudo apt upgrade -y`. We also made sure to install Open-VM-Tools with the command `sudo apt install open-vm-tools-desktop -y && sudo systemctl enable -now open-vm-tools-desktop` to ensure that the virtual machine had no compatibility problems with VMware Workstation. We followed up those commands by running `sudo reboot now` to restart the virtual machine.

After the virtual machine was rebooted, we could proceed with our installation of DVWA. We first grabbed the unofficial installation script from the DVWA GitHub repo and ran it. This script automatically installed DVWA with all of the prerequisites required. After installation was successful, we copied the configuration files by navigating to Apache's web directory, then the DVWA's subdirectory, and then issuing this command `cp config/config.inc.php.dist config/config.inc.php`. We then issued `sudo systemctl restart apache2` to restart DVWA to allow it to restart with the correct configuration files. After the Apache web server was restarted, we navigated to the DVWA web page `http://192.168.99.100/DVWA` and were able to log in using the default credentials of `admin` and `password`. See Figure 3.

2.1.2 Suricata Configuration

Once the installation of DVWA was completed, we proceeded with the installation of Suricata onto our pfSense virtual machine. Suricata's installation was extremely simple. We navigated to the web GUI of pfSense's Package Manager and installed the Suricata package. After the package was installed. We configured Suricata by first enabling the rules we wanted to include. We chose to enable the Emerging Threats open-source ruleset and Snort's free ruleset for registered users. It is important to mention that for Snort's ruleset, we did have to input our license key into the Suricata configuration page. After that was completed, we configured the interface we wanted Suricata to listen to. We chose to have it listen on OPT1, which is the unused network adapter we added within the virtual machine configuration of VMware Workstation. We chose this adapter since Suricata places the adapter in promiscuous mode to listen to traffic on the entire network. Next, we ensured that Suricata was using its IDS mode rather than its IPS mode, as we did not want to start blocking network traffic immediately if we had a misconfiguration. This completed the configuration of Suricata, and we enabled Suricata within that network interface and started the service.

2.1.3 Exploited Attack Vector

After installing Suricata and DVWA, we are almost ready to start seeing if Suricata will detect threats. However, we must first install a Kali Linux virtual machine within our virtualized lab environment. Having a Kali Linux VM will allow us to test a multitude of attacks against DVWA. To install Kali Linux, we must first create a virtual machine within VMware Workstation, then we assigned it 4 CPU cores, 8 GB of RAM, and a 64 GB hard disk. Next, we follow the GUI installation and create a user account. We chose the default choices and continued with the installation. Once Kali was finished installing, we rebooted the system. Once Kali was restarted, we logged in and made sure we had

internet connectivity. Kali grabbed an IP address of `192.168.99.103` from our pfSense DHCP server. We updated all packages using `sudo apt update && sudo apt upgrade -y`. We also made sure to install Open-VM-Tools with the command `sudo apt install open-vm-tools-desktop -y && sudo systemctl enable -now open-vm-tools-desktop` within both our Kali Linux VM and our Ubuntu VM. After those commands were completed, we restarted the Kali Linux VM. At this point, we had everything we needed to start testing Suricata's capabilities. We had the lab environment, the web application server, and the tools necessary to exploit the web server. See Figure 4.

3 Discussion

Our first small attack we ran was a network scan of the DVWA host. This was completed by using the command `nmap -T4 -A -v 192.168.99.100`. After a few seconds, NMAP returned results, and we saw that port 80 was open. This is the port that DVWA was running on. Then, we looked at the alerts section within the Suricata tab on pfSense, and we saw that Suricata had successfully detected the port scan of DVWA. At this point, we were extremely excited as we had proven that our lab environment was correctly configured and was working correctly. See Figure 5.

For our next attack against DVWA, we decided to test the functionality of Suricata in detecting DoS attacks against DVWA. We decided to use the tool HPING3. HPING3 has many different options, but we ultimately chose to use `sudo hping3 -flood -p 80 -rand-source 192.168.99.100`. Almost immediately, we saw a slowdown when trying to access the pfSense website. Opening Wireshark within the Kali VM, we could see the sheer amount of traffic flowing to the DVWA host. When checking the alerts of Suricata, we could clearly see that Suricata had detected the DoS attack from HPING3. See Figure 6 and Figure 7.

For our last attack, we wanted to use an attack that would be commonly used within real-world examples. For this reason, we chose to do a SQL Injection attack. Looking at DVWA's documentation, in order to perform a SQL Injection, we must input this command `?id=a' UNION SELECT "hello","ist815";- -&Submit=Submit #` into the vulnerable search box within the DVWA site. After inputting that command and waiting a few seconds, we realized that no alert was generated even though we had seen alerts for previous attacks. After some troubleshooting, we figured out that we had to do two things. First, we had to make sure that HTTP traffic analysis was enabled within our Suricata configuration for interface OPT1. After changing this setting, we ran the test again and still did not receive an alert. However, after checking the EVE.json file, we saw the SQL injection attack being observed from Suricata. This led us to believe that Suricata and our configured rules did not align with the attack being executed. To solve this problem, we wrote our own rules to detect the attack being used against DVWA. After writing the rules, we ran the attack again and noticed that Suricata detected an alert for our SQL Injection within DVWA. This was a major achievement for this research project. See Figure 8 and Figure 9.

After the previous successful attack, we wanted to adapt the attack into an attack that you would see being utilized out in public. We wanted to stick with SQL Injection, but automate the attack to try for different injection techniques. To do this, we used SQLMAP within our Kali Linux VM. We crafted and issued the command `sqlmap -u "http://192.168.99.100/DVWA/vu1nerabi1ities/sqli/" -data="id=1&Submit" -level=3 -risk=2 -batch`. After switching tabs to the Suricata alerts page, we were glad to see that the attacks were being detected and alerted. We also noticed that since SQLMAP was using common injection techniques from previous CVEs. Our rulesets, ET-OPEN and Snort's Registered-User list, detected the attacks by default with no modification. See Figure 10, Figure 11, and Figure 12.

4 Conclusion

It is important to note that an IDS or IPS is not a catch-all system. Cyberattacks can still occur within your organization's network and bypass the IDS or IPS detection rules. The threat landscape is constantly changing, and the overall intrusion detection process must be kept up to date to battle these attacks. Your organization should not rely on just one defensive measure against cyberattacks. There should be multiple layers of security that give your organization the best possible chance to withstand these intricate cybersecurity events.

References

Agarwal, N., & Hussain, S. Z. (2018a). A closer look at intrusion detection system for web applications. *Security and Communication Networks*, *2018*, 9601357. https://doi.org/10.1155/2018/9601357

Agarwal, N., & Hussain, S. Z. (2018b, May 28). Identification of flaws in the design of signatures for intrusion detection systems. https://doi.org/10.48550/arXiv.1805.10848

Ahmad, A., Hadgkiss, J., & Ruighaver, A. B. (2012). Incident response teams – challenges in supporting the organisational security function. *Computers & Security*, *31*(5), 643–652. https://doi.org/10.1016/j.cose.2012.04.001

Albin, E., & Rowe, N. C. (2012). A realistic experimental comparison of the suricata and snort intrusion-detection systems. *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, 122–127. https://doi.org/10.1109/WAINA.2012.29

Alzahrani, A., Alqazzaz, A., Zhu, Y., Fu, H., & Almashfi, N. (2017). Web application security tools analysis. *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*, 237–242. https://doi.org/10.1109/BigDataSecurity.2017.47

Barrett, M. P. (2018). Framework for improving critical infrastructure cybersecurity version 1.1. *NIST*. Retrieved November 7, 2025, from https://www.nist.gov/publications/framework-improving-critical-infrastructure-cybersecurity-version-11

Boukebous, A. A. E., Fettache, M. I., Bendiab, G., & Shiaeles, S. (2023). A comparative analysis of snort 3 and suricata. *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, 1–6. https://doi.org/10.1109/GlobConET56651.2023.10150141

Day, D. J., & Burns, B. M. (2011). A performance analysis of snort and suricata network intrusion detection and prevention engines. https://personales.upv.es/thinkmind/dl/conferences/icds/icds_2011/icds_2011_7_40_90007.pdf

Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., & Madinabeitia, G. (2022). On the detection capabilities of signature-based intrusion detection systems in the context of web attacks. *Applied Sciences*, *12*(2), 852. https://doi.org/10.3390/app12020852

Einy, S., Oz, C., & Navaei, Y. D. (2021). The anomaly and signature-based IDS for network security using hybrid inference systems. *Mathematical Problems in Engineering*, *2021*, 6639714. https://doi.org/10.1155/2021/6639714

Fekolkin, R. (2015). Intrusion detection and prevention systems: Overview of snort and suricata. Retrieved November 7, 2025, from https://www.researchgate.net/publication/297171228_Intrusion_Detection_and_Prevention_Systems_Overview_of_Snort_and_Suricata

Hylender, C. D., Langlois, P., Pinto, A., & Widup, S. (2025). *2025 data breach investigations report*. Verizon Business. Retrieved October 10, 2025, from https://www.verizon.com/business/resources/T4a/reports/2025-dbir-data-breach-investigations-report.pdf

International Organization for Standardization. (2022a). *Information security, cybersecurity and privacy protection — information security controls* (ISO/IEC 27001:2022). https://www.iso.org/standard/82875.html

International Organization for Standardization. (2022b). *Information security, cybersecurity and privacy protection — information security management systems — requirements* (ISO/IEC 27002:2022). https://www.iso.org/standard/75652.html

Khan, S., & Motwani, D. (2017). Implementation of IDS for web application attack using evolutionary algorithm. *2017 International Conference on Intelligent Computing and Control (I2C2)*, 1–5. https://doi.org/10.1109/I2C2.2017.8321956

Leblond, E. (2021, June 7). *Suricata: The first 12 years of innovation*. Stamus Networks. Retrieved November 15, 2025, from https://www.stamus-networks.com/blog/suricata-the-first-12-years-of-innovation

Liñan-Acosta, B., Bazalar-Gonzales, O., & Santisteban, J. (2025). Frameworks for cyberattack prevention: A comparative analysis. *2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, 1–5. https://doi.org/10.1109/ACDSA65407.2025.11166447

Manev, P. (2021). Scaling suricata for enterprise deployment. *Stamus Networks*. Retrieved November 15, 2025, from https://www.stamus-networks.com/hubfs/Whitepapers/StamusNetworks-WP-ScalingSuri-092021-1.pdf

National Institute of Standards and Technology. (2024, February 26). *The NIST cybersecurity framework (CSF) 2.0* (NIST CSWP 29). National Institute of Standards and Technology. Gaithersburg, MD. https://doi.org/10.6028/NIST.CSWP.29

Open Information Security Foundation. (2024, November 13). *Our story*. Suricata. Retrieved November 15, 2025, from https://suricata.io/our-story/

Open Information Security Foundation. (2025, November 6). *Suricata's latest features*. Suricata. Retrieved November 15, 2025, from https://suricata.io/features/

Raharjo, D. H. K., Nurmala, A., Pambudi, R. D., & Sari, R. F. (2022). Performance evaluation of intrusion detection system performance for traffic anomaly detection based on active IP reputation rules. *2022 3rd International Conference on Electrical Engineering and Informatics (ICon EEI)*, 75–79. https://doi.org/10.1109/IConEEI55709.2022.9972298

Robinette, D. (2024, January 15). *Is suricata an IPS or IDS?* Stamus Networks. Retrieved November 15, 2025, from https://www.stamus-networks.com/blog/is-suricata-an-ips-or-ids

RS, G. (2024, August 8). *Guide to suricata: Network security, IDS, IPS, and NSM*. Devzery. Retrieved November 15, 2025, from https://www.devzery.com/post/guide-to-suricata-network-security-ids-ips-and-nsm

Scarfone, K., & Mell, P. (2007, February 20). *Guide to intrusion detection and prevention systems (IDPS)* (NIST Special Publication (SP) 800-94). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-94

Snort Team. (2025). *Network intrusion detection & prevention system*. Snort. Retrieved November 15, 2025, from https://www.snort.org/

Stock, A. v. d., Glas, B., Smithline, N., & Gigler, T. (2021). *OWASP top ten*. OWASP Foundation. Retrieved November 7, 2025, from https://owasp.org/www-project-top-ten/

Taqiyuddin Ali, M. (2025). Evaluation of ISO/IEC 27001 framework implementation for information security in organizations: A systematic literature review. Retrieved November 7, 2025, from https://www.researchgate.net/publication/393090821_Evaluation_of_ISOIEC_27001_Framework_Implementation_for_Information_Security_in_Organizations_A_Systematic_Literature_Review

The Zeek Project. (2020, February 20). *About zeek*. Zeek. Retrieved November 15, 2025, from https://zeek.org/about/

Thongkanchorn, K., Ngamsuriyaroj, S., & Visoottiviseth, V. (2013). Evaluation studies of three intrusion detection systems under various attacks and rule sets. *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*, 1–4. https://doi.org/10.1109/TENCON.2013.6718975

Waleed, A., Jamali, A. F., & Masood, A. (2022). Which open-source IDS? snort, suricata or zeek. *Computer Networks*, *213*, 109116. https://doi.org/10.1016/j.comnet.2022.109116

White, J. S., Fitzsimmons, T., & Matthews, J. N. (2013). Quantitative analysis of intrusion detection systems: Snort and suricata. *Cyber Sensing 2013*, *8757*, 10–21. https://doi.org/10.1117/12.2015616

Wojtas, G. (2025, November 6). *Suricata deep dive: What it is, how it works, and why it matters*. Retrieved November 15, 2025, from https://library.nagios.com/techtips/suricata-deep-dive-what-why-how/

Wood, R. (2025, November 15). *Damn Vulnerable Web Application (DVWA)*. Retrieved November 15, 2025, from https://github.com/digininja/DVWA

A Background

An IDS is a software or service that can automate the manual intrusion detection process (Scarfone & Mell, 2007). It has the ability to monitor and identify potentially harmful packets that are trying to gain access to a specific web application or server (Scarfone & Mell, 2007). The IDS then usually alerts the user of these identified incidents so that they can remedy them. The alerts can be configured to output to numerous different types of platforms or services, including being stored within the syslog file of the specific IDS that you are using. Using this method allows for different logging services to interact with that syslog file and import it into other platforms (Scarfone & Mell, 2007). Commonly, today we see the output of an IDS ingested into a Security Information and Event Management (SIEM) system. This allows the cybersecurity professional to have a centralized location to see all of the alerts within a potential organization (Scarfone & Mell, 2007). However, to assemble and create a log for cybersecurity professionals, the IDS must collect information from different areas of the web application or server.

The information that is collected and stored within a log file varies due to what service the IDS is protecting. The log file typically contains the date and time of when the alert occurred, the hostname of the IDS service that is running, the IP address of both the source of the packet and the destination of that packet, the port for both the source and destination IP addresses, and the designated threat level of that attack. You might also see the protocol that is being used to launch an attack. For example, if an HTTP web server is being attacked, you will see the HTTP protocol listed within the log file. It is important to note that this output is non-standardized and can change between different IDSs. A cybersecurity professional could also change many aspects of the output log file to match their needs. In a lot of production environments today, we see the MITRE ATT&CK framework being added to the log file. This addition helps the cybersecurity professional categorize the incident so that they respond to more urgent incidents first. While an IDS detects incidents manually, an Intrusion Prevention System (IPS) can alleviate some pressure on the responding cybersecurity professional.

An Intrusion Prevention System (IPS), like an IDS, is a software or service that can automate the manual intrusion detection process, but unlike an IDS, the IPS attempts to prevent the attack from being executed (Scarfone & Mell, 2007). Scarfone and Mell (2007) state that this could be done in three ways: “terminate the network connection or user session that is being used for the attack, block access to the target (or possibly other likely targets) from the offending user account, IP address, or other attacker attribute or block all access to the targeted host, service, application, or other resource” (p. 17). These automatic prevention methods will attempt to stop the attack or incident from happening and potentially block the attacker’s source IP address so that they cannot attempt the attack again (Scarfone & Mell, 2007). Since an IPS builds on the foundational principles of the intrusion detection process, it is important to understand the detection methods that enable these services to function.

There are three primary detection methods used within an intrusion detection system to detect these cyberattacks: Signature-Based Detection, Anomaly-Based Detection, and Stateful Protocol Analysis (Scarfone & Mell, 2007). Signature-Based Detection, like the name suggests, uses a signature to identify and categorize potential incidents. A signature is defined as “a pattern that corresponds to a known threat” (Scarfone & Mell, 2007, p. 18). Signature-Based Detection is extremely useful in a scenario where the attack vector has already been established and detected before. An IDS or an IPS can pick up on the attack right away. However, a Signature-Based Detection does have its downfalls. For example, if your organization is facing a tailor-made payload, then Signature-Based Detection will have a difficult time recognizing it as the signature will be something that it has never seen before. The Anomaly-Based Detection profiles the typical traffic of a web application or server. Then, once an irregular event is detected, an alert is issued. Unlike Signature-Based Detection, Anomaly-Based Detection is extremely good at detecting unknown threats. However, this detection model can still have some downsides. For example, if you experience any sort of unusual traffic that is not malicious in nature, it can be detected or even blocked. This could prevent end-users from accessing the resources they need. Within different IDS or IPS systems, Stateful Protocol Analysis can be referred to by different names. Some intrusion detection systems might refer to it as Mixed Methods Detection or Deep Packet Inspection (Scarfone & Mell, 2007). The premise of this detection method is that it identifies what the user is trying to achieve and ensures that the user’s task does not go beyond the required scope for the operation. For example, within a File Transfer Protocol (FTP) session, a user can transfer files between two different computers or servers. However, when

transferring, the user must authenticate with the receiving machine. If the user attempts to transfer a file within unauthenticated sessions, the IDS or IPS will detect it as unusual behavior. It is important to recognize that the different types of IDS or IPSs can change how these irregular events can be detected.

There are multiple different ways an IDS or an IPS can be deployed within your organization's network, including: Network-Based, Wireless-Based, Network Behavior Analysis (NBA), and Host-Based (Scarfone & Mell, 2007). A Network-Based intrusion detection system captures network traffic moving within your network and analyzes it using detection methods. This type of device is extremely useful when you are deploying a firewall and need a general layer of security. Usually, other intrusion detection systems will be placed within your network. A Wireless intrusion detection system, which is not used extensively today, monitors wireless traffic from Access Points (APs). In most cases, this category of device is replaced with a Network-Based intrusion detection system. NBAs are a category of device similar to a Network-Based IDS, but they are specialized in dealing with large and unusual traffic flows (Scarfone & Mell, 2007). Lastly, Host-Based IDSs are deployed within the web application servers to ensure that threats are detected and potentially prevented from interfering with end-user abilities. It is important to note that Network-Based and NBAs are usually placed at the firewall within a network topology, whereas a Wireless intrusion detection system is deployed within a wireless controller, and Host-Based are deployed within the device you want to protect.

When opting to implement an IDS or an IPS, you must first choose what brand of intrusion detection system you want. There are multiple prominent IDSs on the market today. For enterprise solutions, your organization might choose a solution from Palo Alto Networks or Cisco. For smaller-scale networks, your organization might choose Snort or Suricata. It is important to note that enterprise-scale offerings often use Snort or Suricata under the hood but have proprietary rulesets that give them a market edge. For personal use, you might choose Fail2Ban. Choosing the right solution is crucial, and you will have to evaluate your organization's needs to pick the right one. In this research project, we will be exploring Suricata as it will allow us to learn more about IDS and IPS systems in production environments.

B Related Work

To achieve a deeper understanding of Intrusion Detection and Prevention Systems (IDPS), we must conduct a literature review to examine previous research on this topic. This literature review examines prior research on IDPS technologies, specifically Suricata, and their compliance with regulatory frameworks and web application security best practices. This literature review will also look at the inner workings of Suricata and the open-source and paid ruleset that make Suricata extremely powerful against new cybersecurity threats. This literature review will help this research project achieve the goal of evaluating how Suricata functions in real-world environments and assessing its advantages and limitations within those environments.

B.1 Regulatory Standards

B.1.1 NIST Cybersecurity Framework (CSF)

The National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) provides a structured approach for mitigating cybersecurity threats within various organizations (Barrett, 2018; National Institute of Standards and Technology, 2024). The CSF defines five core functions: Identify, Protect, Detect, Respond, and Recover (National Institute of Standards and Technology, 2024). These functions help establish a lifecycle for effective cybersecurity management (National Institute of Standards and Technology, 2024). Within this structure, IDPSs, such as Suricata, support the Detect and Respond phases. This is due to them being able to monitor traffic for attack vectors and generate alerts for potential attacks (Fekolkin, 2015; Khan & Motwani, 2017; Scarfone & Mell, 2007). This fulfillment of these functions gives the organization a definitive answer whether or not an IDPS would be suitable for them within their network environment.

B.1.2 ISO/IEC 27001 and 27002

The ISO/IEC 27001 and 27002 standards define best practices for implementing an Information Security Management System (ISMS) within organizations (International Organization for Standardization, 2022a, 2022b). These standards showcase the importance of continuous monitoring, logging, and event management, especially when dealing with important organizational data ((International Organization for Standardization, 2022a, 2022b; Liñan-Acosta et al., 2025). Suricata can support these actions through its alerting capability and due to its output within EVE.JSON. This file provides a way for the logfile to be ingested into a Security Information and Event Management (SIEM) or a data visualization platform like Grafana. Waleed et al. (2022) found that implementing Suricata within a network environment can help the organization detect and monitor threats and network trends. Suricata can serve as both a technical safeguard to meet ISO compliance within an organization's ISMS (International Organization for Standardization, 2022a, 2022b; Taqiyuddin Ali, 2025; Thongkanchorn et al., 2013).

B.1.3 OWASP Framework

The Open Web Application Security Project (OWASP) defines standards for identifying and mitigating the most critical web application vulnerabilities (Stock et al., 2021). This is defined within OWASP's Top 10 list (Stock et al., 2021). It is important to note that while OWASP primarily provides information for software developers, the framework still provides valuable information when designing detection rules for a Suricata implementation (Agarwal & Hussain, 2018a; Stock et al., 2021). The Top 10 list provides an easy way to identify the most critical attack vectors to monitor within a web application. Research by Díaz-Verdejo et al. (2022) and Agarwal and Hussain (2018a, 2018b) indicates that using the OWASP vulnerability list within Suricata's detection rules can improve detection coverage and reduce false negatives. By doing this, Suricata can further protect web application servers from the growing cybersecurity threats.

B.2 Intrusion Detection and Prevention Systems (IDPS)

B.2.1 Suricata Architecture

Suricata was developed as an open-source, multi-threaded IDS/IPS to help identify and potentially block threats within a network environment (Waleed et al., 2022). Some previous research studies showcase that Suricata often outperforms similar services such as Snort, Zeek/Bro, and Snort (Boukebous et al., 2023; Day & Burns, 2011; Waleed et al., 2022). This is especially true in scenarios where the network is experiencing heavy throughput (Albin & Rowe, 2012). This makes Suricata an excellent choice when looking to implement an IDPS for a web application server, as it will not be overloaded when network traffic hits all-time highs.

Additionally, Suricata supports parsing of specific network protocols such as HTTP, TLS, and DNS (Agarwal & Hussain, 2018a; White et al., 2013). However, the detection accuracy of Suricata heavily depends on the configuration and the rules assigned (Albin & Rowe, 2012; Díaz-Verdejo et al., 2022; Einy et al., 2021; Raharjo et al., 2022; Thongkanchorn et al., 2013). Albin and Rowe (2012) conducted a research study showcasing that Suricata achieved a higher accuracy in detecting malicious traffic over time compared to Snort. However, Snort maintained a higher network traffic flow compared to Suricata (Albin & Rowe, 2012). Research findings like these showcase the capabilities of Suricata when configured correctly. It is important to note that Suricata is not supposed to be the only line of defense for web application servers.

B.2.2 Rulesets' Management

Suricata's performance and detection accuracy heavily rely on the implemented and defined rulesets (Agarwal & Hussain, 2018b; Day & Burns, 2011; Raharjo et al., 2022). It is important to note that by default, Suricata has very limited abilities in detecting malicious threats, especially if they are new to the cybersecurity industry. Agarwal and Hussain (2018b) state that incomplete or redundant rulesets can result in false positives and potentially undetected cyberattacks. Díaz-Verdejo et al. (2022) found that using the Emerging Threats (ET) open-source rules may be able to detect complex web application attacks. This problem alludes to the idea of using multiple rulesets, oftentimes from different vendors. However, Raharjo et al. (2022) showcased that implementing multiple rules may increase detection rates, but it also reduces Suricata's performance overall. When Suricata's

performance decreases, the IDPS might not be able to keep up and will skip potentially malicious packets. Being able to maintain the balance between detection accuracy and performance is an extremely difficult task to manage for organizations and for security researchers.

B.3 Web Application Security

The cybersecurity industry is constantly evolving to protect network infrastructure against cyberattacks (Ahmad et al., 2012). However, new cyberthreats are being developed, and web application servers continue to be valuable targets for threat actors (Hylender et al., 2025). This is due to many websites still continuing to use outdated and legacy code within important components (Alzahrani et al., 2017; Díaz-Verdejo et al., 2022; Khan & Motwani, 2017). For example, attack vectors such as SQL injection, cross-site scripting, and command injection are still widely exploited in production environments today, even though the exploits have been around since the dawn of the internet (Stock et al., 2021). These exploits are also some of the common techniques threat actors use in order to gain access to a web application (Stock et al., 2021). This is why Suricata can be used for the protection of web application servers, as it can inspect the HTTP or HTTPS packets to ensure that these techniques are not being used.

B.4 Overview of Existing Research

This literature review showcases the importance of Suricata being evaluated with cybersecurity frameworks such as NIST, ISO/IEC, and OWASP standards. Additionally, it allows organizations to implement a proven IDPS within their production network environment to prevent cyberattacks from occurring on their web application servers. Suricata's open-source architecture provides a proven IDPS for organizations that want to implement a layered defense strategy. However, Suricata does not come without its challenges, such as the ruleset performance degradation. Despite its proven capabilities, this research project's goal is to evaluate Suricata's capability in protecting vulnerable web application servers from common attack vectors.

C Suricata

As web applications continue to be a primary target for attackers, there is a growing need to evaluate effective tools that can detect and prevent exploits at the network level. In order to stop them before they can reach vulnerable web servers. We chose Suricata as the main focus for this research project for this reason. Suricata is an open-source IDS/IPS system that is designed to protect network infrastructure from growing cyberthreats (Open Information Security Foundation, 2025). Suricata is often chosen within enterprise environments for its speed, flexibility, and Deep Packet Inspection capabilities, which make it perfect for detecting and preventing attacks against web application servers (Open Information Security Foundation, 2025). Suricata also has the ability to perform multi-threaded packet analysis, protocol identification, and real-time intrusion prevention (Open Information Security Foundation, 2025). This allows for alerting of Network-Based threats in our simulated lab environment (Open Information Security Foundation, 2025).

The motivation behind this project is to evaluate Suricata's ability to protect web application servers against common attack vectors. Attacks such as SQL Injection attacks, Cross-Site Scripting (XSS), and Distributed Denial of Service (DDoS) attacks all play a significant role in today's cybersecurity landscape (Hylender et al., 2025). This research paper ultimately aims to measure how well Suricata performs in detecting, logging, and mitigating these threats. This will enable us to better understand open-source security tools with regard to modern web application servers. By implementing Suricata in a controlled, real-world lab environment, this research paper seeks to assess its detection accuracy, performance efficiency, and practical usability.

C.1 History

Suricata was developed as an open-source next-generation IDS/IPS system under the support of the Open Information Security Foundation (OISF) (Leblond, 2021; Open Information Security Foundation, 2024, 2025). The OISF is a non-profit organization dedicated to improving the cybersecurity industry's network security monitoring and intrusion prevention techniques (Open Information Security Foundation, 2024). According to the OSIF's historical timeline, prototype work began around

2007 and early 2008, before the first public release of Suricata 1.0 in 2010 (Open Information Security Foundation, 2024). This was when OISF was formally founded (Open Information Security Foundation, 2024). OISF was initially funded in part by the U.S. Department of Homeland Security (DHS) and by OISF consortium members (Open Information Security Foundation, 2024). Suricata's and OSIF's mission is to always remain free and open source under GPL v2 licensing (Open Information Security Foundation, 2024). This was made possible by the community involvement and vendor support (Leblond, 2021; Open Information Security Foundation, 2024). Over the years, Suricata has advanced to support multithreading, file extraction, Deep Packet Inspection (DPI), and inline IPS modes (Open Information Security Foundation, 2025). Suricata has evolved from purely detection to a complete network security monitoring (NSM) system (Leblond, 2021).

C.2 Other Enterprise IDS/IPS Solutions

There are several other open-source tools that are widely used in enterprise environments, such as Snort 3 and Zeek. Snort is one of the most well-known IDS/IPS engines. It has an extensive rule base and integrates with Cisco's Talos threat intelligence feeds, which is one of its leading features (Snort Team, 2025). It supports both inline blocking and passive detection, allowing it to be integrated into various types of network environments (Snort Team, 2025). Due to it being around for a long time and having a large community, it benefits from frequent rule updates to keep up with modern threats (Snort Team, 2025). This allows Snort to still be used as a Signature-Based detection engine (Snort Team, 2025). Zeek takes a different approach. Rather than a Signature-Based IDS, Zeek primarily functions as an NSM system (The Zeek Project, 2020). Instead of flagging threats through predefined signatures, Zeek analyzes how services behave on the network (The Zeek Project, 2020). This allows Zeek to be well-suited for detecting threats that a Signature-Based IDS/IPS would miss (The Zeek Project, 2020). These other open-source tools can complement or substitute for Suricata, depending on whether the use case prioritizes Deep Packet Inspection or centralized analytics. However, when used as a basic IDS/IPS, they all function similarly.

C.3 Detection Capabilities

One of the main reasons why you choose to implement Suricata is its vast detection capabilities. Suricata is able to combine Signature-Based detection with automatic protocol identification and Deep Packet Inspection (DPI) (Manev, 2021; Open Information Security Foundation, 2025). This allows Suricata to support a broad range of protocols, including HTTP, TLS, DNS, and SMB (Open Information Security Foundation, 2025). Snort, however, retains its strong Rule-Based detection by using Cisco Talos's threat-intelligence feeds (Snort Team, 2025). These feeds are widely used and are an industry standard in enterprise environments (Snort Team, 2025). Zeek takes a different approach. Zeek uses Event-Driven Analysis rather than signatures (The Zeek Project, 2020). This provides behavioral insight by recording detailed metadata about network sessions (The Zeek Project, 2020). This is a major selling point of Zeek, which can help with anomaly detection and forensic correlation (The Zeek Project, 2020).

C.4 Operational Complexity

Deploying Suricata at scale requires a lot of insight into how an organization will be using Suricata within their daily operations. Although Suricata is open-source, it requires skilled network administrators and a significant time investment to implement correctly (Manev, 2021). Snort is easier to maintain due to Cisco's great documentation (Snort Team, 2025). Snort can also be managed through Cisco's Firepower Management Center (Snort Team, 2025). This allows for complete control over every Snort deployment and allows for automatic rule updates (Snort Team, 2025). Zeek's deployment is more complex due to its event-scripting language and log outputs (The Zeek Project, 2020). These outputs must be stored and categorized within a backend storage system such as Elasticsearch or Splunk (Manev, 2021; The Zeek Project, 2020).

C.5 Recommended Deployment Scenarios

Choosing which IDS/IPS engine to implement in an enterprise environment is an important choice to make. Suricata is best suited for organizations managing high amounts of traffic or requiring low-latency processing (Manev, 2021; Open Information Security Foundation, 2025). It is a strong

choice when inline intrusion prevention, JSON telemetry export, and strong protocol parsing are needed (Manev, 2021; Open Information Security Foundation, 2025). However, Snort is an optimal choice for enterprises that already rely on Cisco infrastructure (Snort Team, 2025). Zeek excels in environments prioritizing forensic analysis, network behavior analytics, and long-term traffic monitoring (The Zeek Project, 2020).

D Comparative Analysis

In our Suricata lab, we are looking to make use of both Suricata's intrusion detection and intrusion prevention services (IDS/IPS). While these services may seem like they perform the same functions, the main difference is that Suricata's IDS is far more passive in its monitoring than its IPS (Robinette, 2024). Having an IDS or an IPS separate from one another can be useful in specific scenarios. In our lab environment demonstration, we aim to showcase why that is the case and how these two technologies can be used within a real-world environment.

In IDS mode, Suricata is able to monitor traffic coming in and out, and it compares traffic to pre-defined rules and threats to determine if it is suspicious (Robinette, 2024; RS, 2024). While the IDS will not actually prevent suspicious traffic, it will issue a clear alert that allows users to take action (Robinette, 2024). Users are also able to create their own rules that can allow for specific alerts (Robinette, 2024). Even though the IDS system is unable to react to suspicious data, there are several advantages that it offers (Robinette, 2024). These include minimum network usage and more visibility due to wider network monitoring (Robinette, 2024). Also, "in some regulations or security policies, actively interfering with network traffic might be restricted" (Robinette, 2024). For this reason, the non-intrusive aspect of passive monitoring offers could help address compliance concerns (Robinette, 2024). While Suricata's IDS system can provide many advantages, sometimes a little bit of automated help can make the incident response process smoother, and this is where Suricata's IPS system can provide a solution (Robinette, 2024; Wojtas, 2025).

Suricata's IPS system offers a more active solution when it scans suspicious/malicious traffic, making use of signature and anomaly-based detection, along with deep packet inspection to dive into the actual contents of data packets (Robinette, 2024; RS, 2024). Unlike the IDS system, "Suricata is deployed inline within the network, meaning that all traffic passes through it before reaching its destination", and can block/allow traffic within the network (RS, 2024). These functions can be based on pre-configured rules, along with user-created rules as well (Robinette, 2024; RS, 2024; Wojtas, 2025). The active monitoring that the IPS system allows for has many advantages over the IDS system. The most important of these advantages is the ability to drop malicious packets if need be (Robinette, 2024; RS, 2024). Some other advantages include applying a limit to the rate of traffic traveling to an IP address, preventing possible DDoS attacks, along with resetting connections that may be transmitting suspicious activity (Robinette, 2024). Despite these advantages, the IPS also contains a couple of disadvantages (Wojtas, 2025). These include how the inline deployment could lead to a break in the network if something goes wrong, as well as the possibility of overwhelming the logging system to the point of being unable to find the most urgent errors (Wojtas, 2025).

Both the IDS and IPS functionality of Suricata are needed in order to fully secure a network. While the IDS can provide substantial transparency into possible malicious data and suspicious patterns, the IPS is needed in order to automate some of the scanning and dropping, which could be mismanaged otherwise due to human error (Robinette, 2024; RS, 2024; Wojtas, 2025). The IPS can help reset connections by transmitting suspicious information, while the IDS could send alerts that allow users to see which IPs are transmitting the packets (Robinette, 2024; RS, 2024). A big part of what we hope to show from our demo is how both the IDS/IPS systems Suricata offers can help secure networks in different ways.

E Supplemental Figures

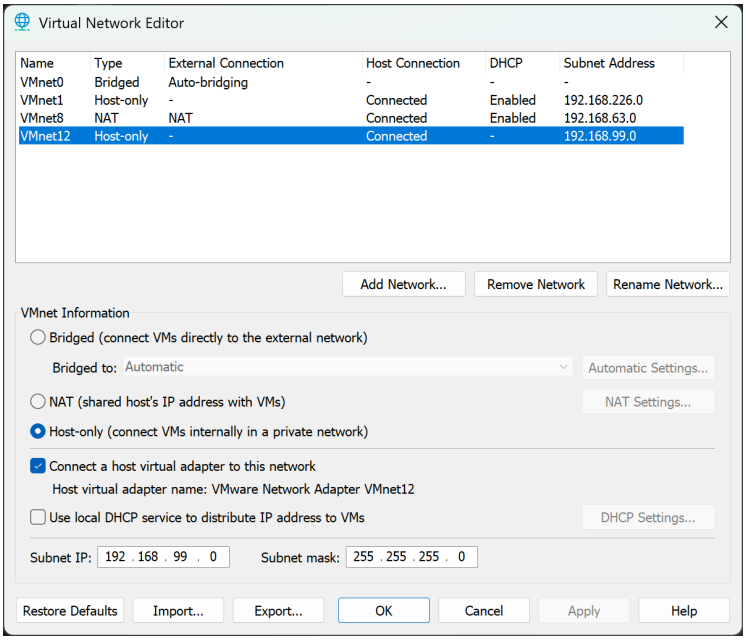

Figure 1: Network adapter settings for VMNet 12 within VMware Workstation

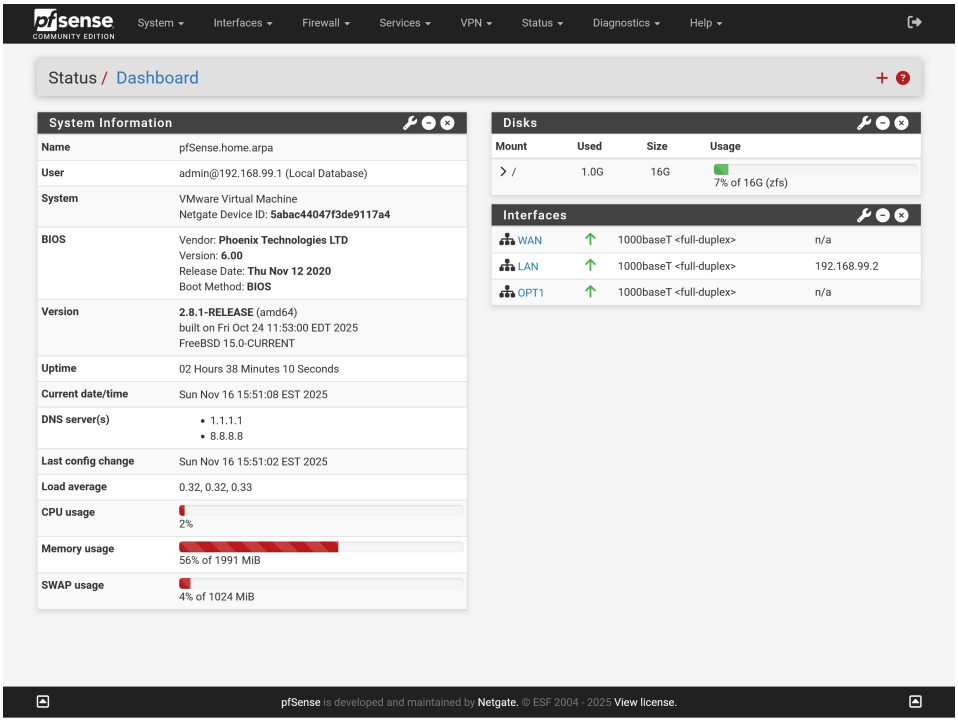

Figure 2: Dashboard view of the pfSense VM

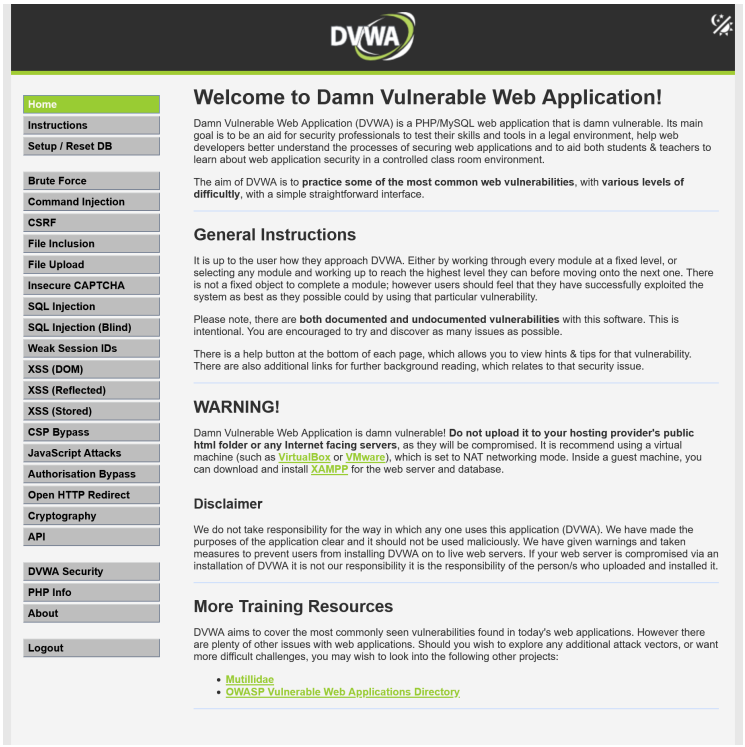

Figure 3: Homepage view for the DVWA web application

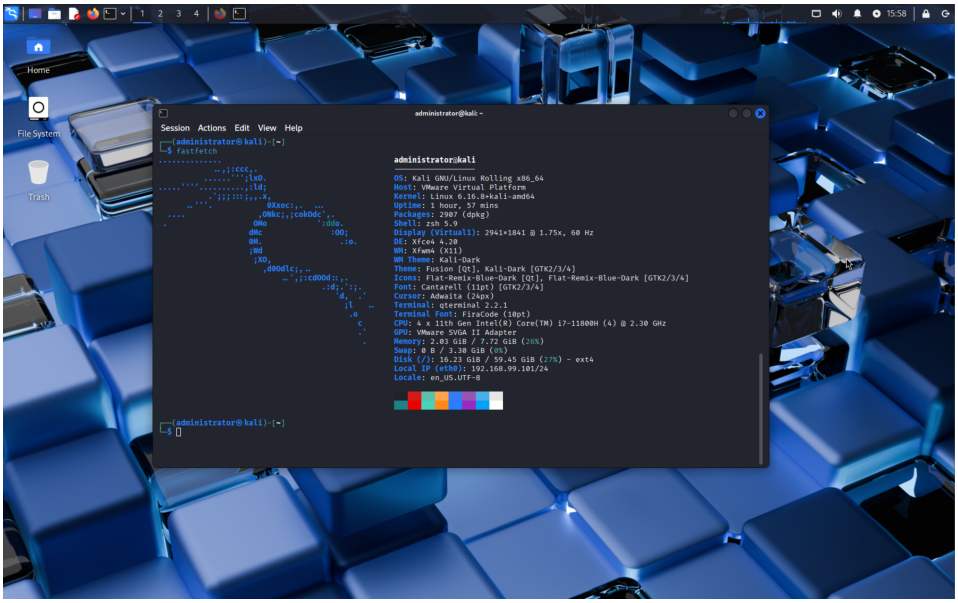

Figure 4: Desktop view of the Kali Linux VM

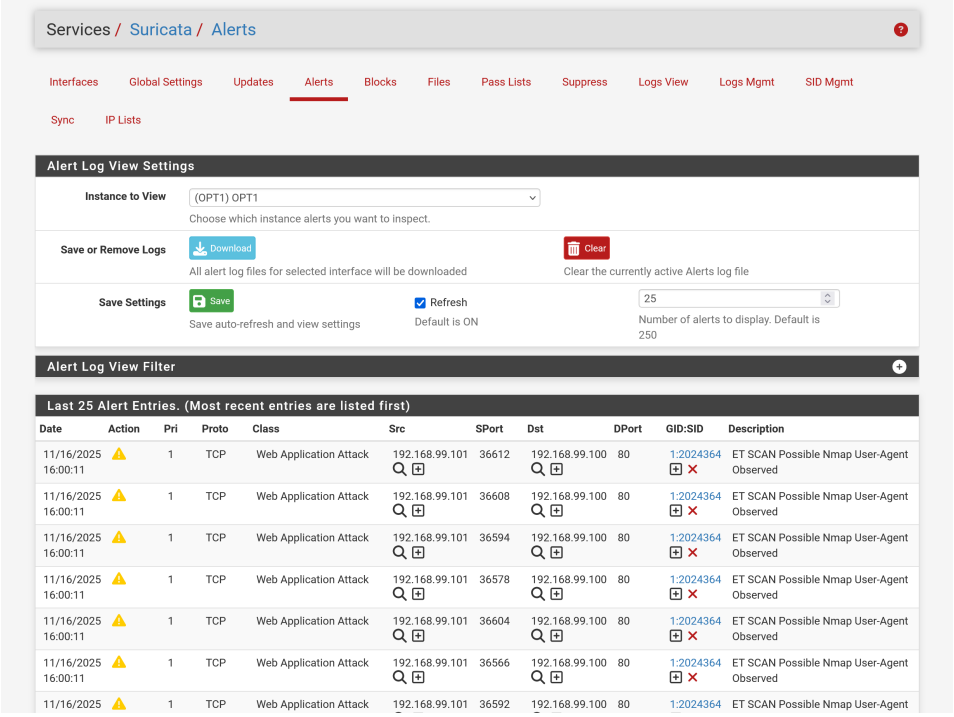

Figure 5: Alerts from Suricata for an NMAP scan

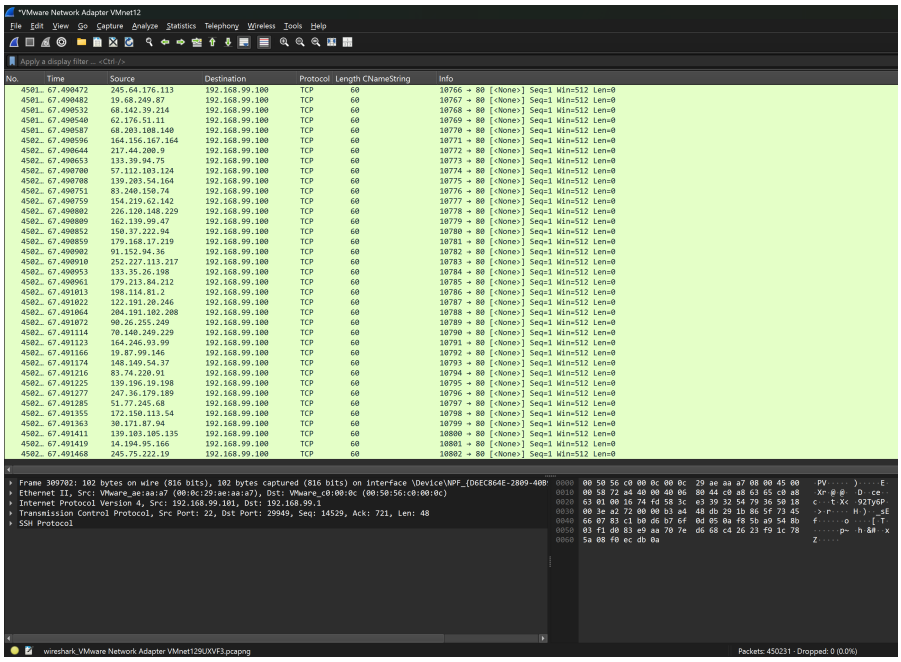

Figure 6: Wireshark capture of a DoS attack from HPING3

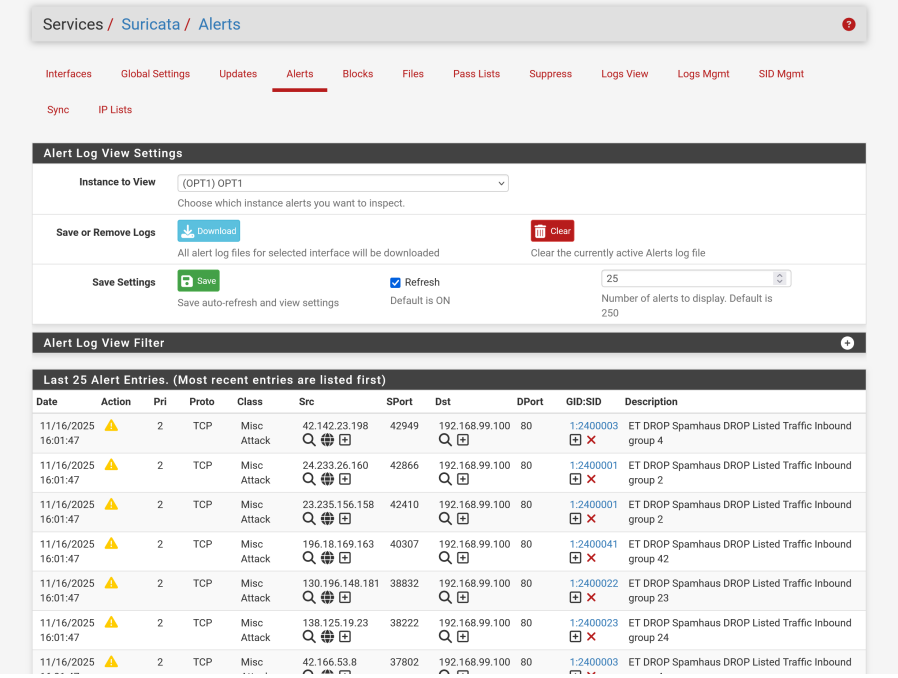

Figure 7: Alerts from Suricata for a DoS attack from HPING3

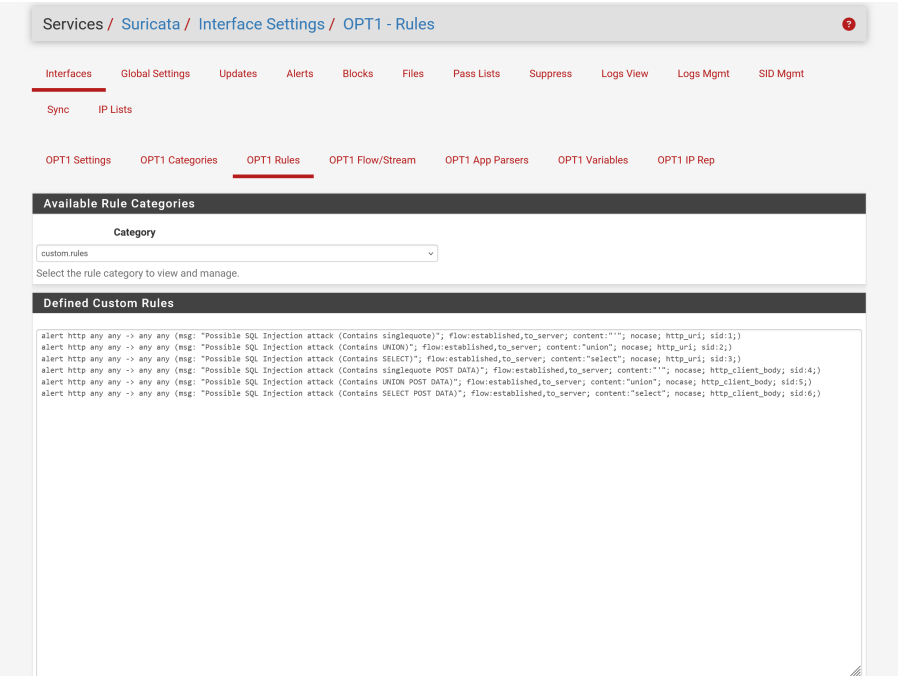

Figure 8: Custom rules defined within Suricata for detecting a custom SQL Injection attack

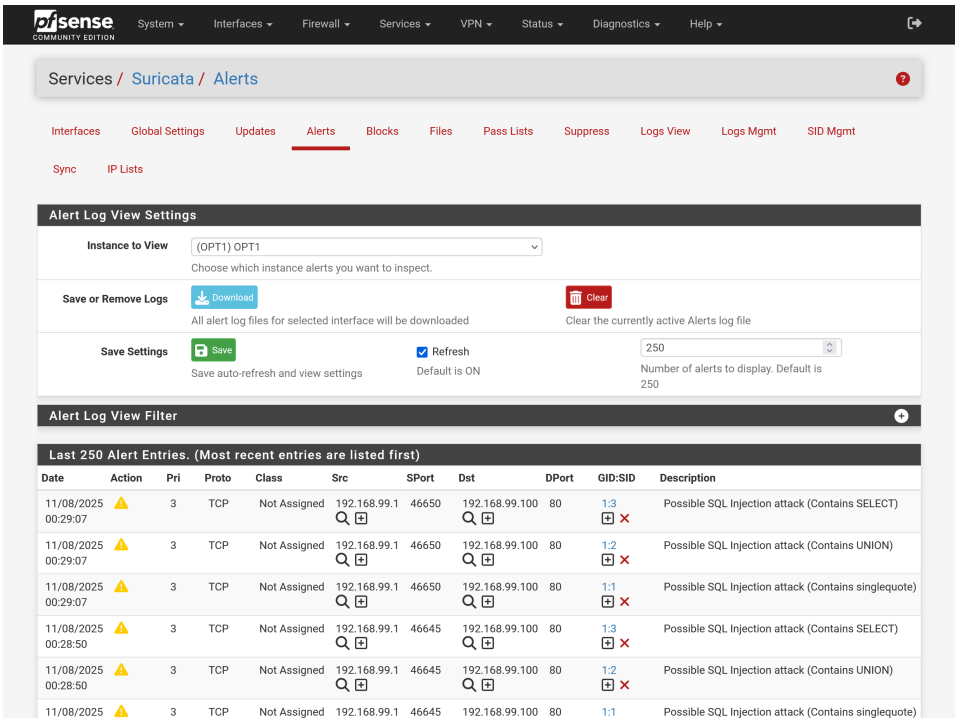

Figure 9: Alerts from Suricata for a manual SQL Injection attack

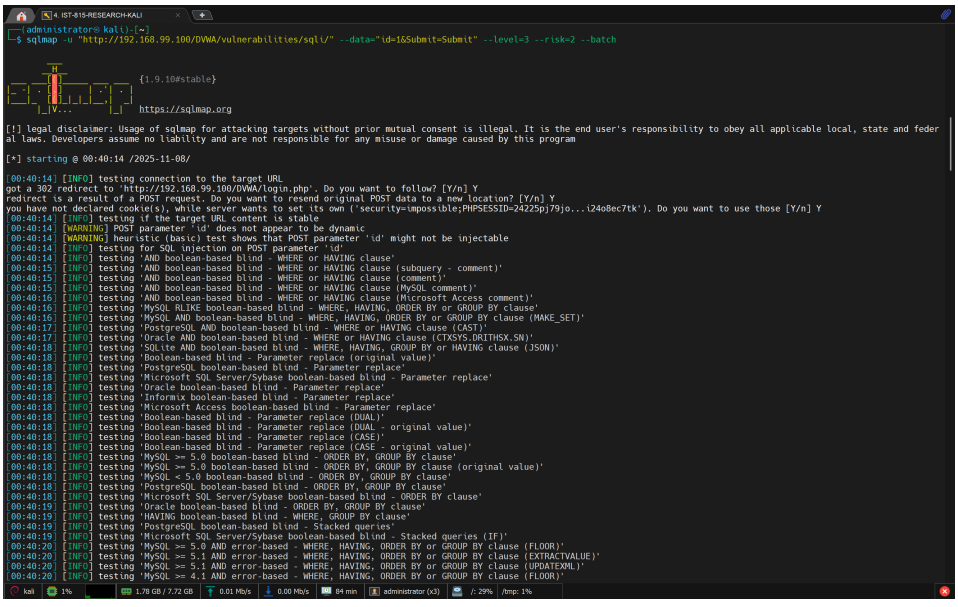

Figure 10: Terminal view of SQLMAP running an automated SQL Injection attack

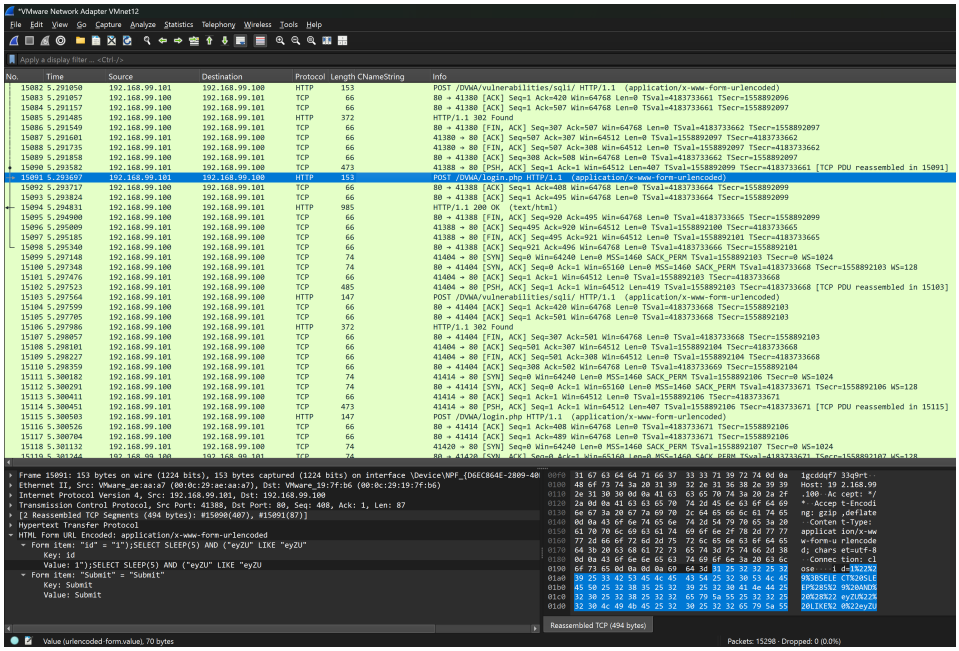

Figure 11: Wireshark capture of an automated SQL Injection attack

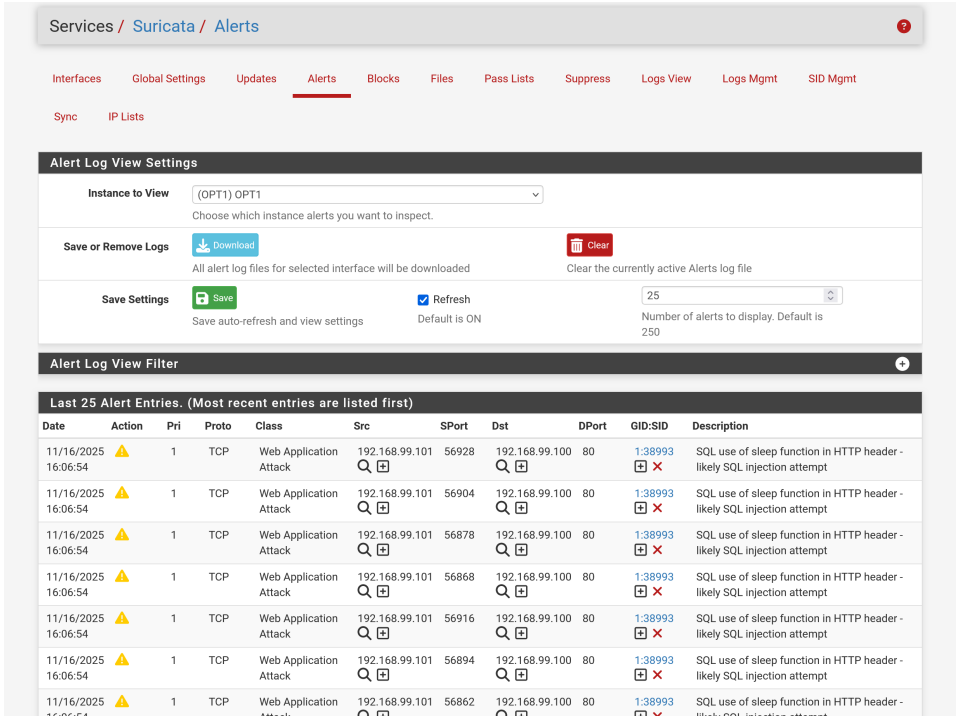

Figure 12: Alerts from Suricata for an automated SQL Injection attack

F Windows Lab Environment Testing

In order to test Suricata installation on a Windows machine as opposed to a Linux machine, we directly installed Suricata 8.0.1 onto a Windows laptop. To do this, we downloaded the latest versions of both Npcap and Suricata. We then followed the installer instructions for both. Once both were installed, we updated the `suricata.yaml` file with the correct network address of the target device and installed all the necessary rulesets, including Emerging Threats ETOpen and Snort Subscriber Free. To install ETOpen, we navigated to the Emerging Threats website and downloaded the latest version of the ruleset. To install Snort's free ruleset, we had to first register for an account, and then we could download the latest version.

After the configuration of Suricata was complete, we proceeded with the preliminary testing of Suricata's IDS abilities. For this Suricata test within the Windows environment, we wanted to see if everything was set up correctly first. To do this, we decided to simulate an ICMP flood using `Nping`. `Nping` allowed us to simulate a small flood where no lasting damage would actually happen to the target device. However, `Nping` would trigger the alerts within Suricata as if it were an actual attack. We conducted this attack from a Ubuntu Linux VM set up on the Windows device. We started Suricata using the command `suricata -c suricata.yaml -i 192.168.89.15 -l log` on the Windows Command Prompt. Next, on the Ubuntu Linux terminal, we initiated the attack using the command `nping -icmp -rate 1000 -count 400 192.168.89.15`. Once we ran the attack, we checked the `alerts.json` file on the Windows device, and we could see that Suricata had detected the attack.

G IPS Lab Environment Testing

In this section, we look at Suricata's capabilities as an IPS, specifically in a virtualized environment. We will use VMWare Workstation Pro 17 as our virtualization platform to conduct the attack testing. We opted to use a Ubuntu VM as the Suricata host machine and a Windows 10 VM as the victim. This will allow us to attack the Windows VM with a Kali Linux VM using common attack methods, such as ping sweep or even port scanning. The experiment will be conducted using three virtual machines running within VMware Workstation Pro 17, where we have an Ubuntu VM configured as the Suricata IPS host, a Kali Linux machine acting as the attacker, and a Windows VM acting as the victim machine.

The three VMs will be placed in the same virtual network, VMNet 8. This will enable Suricata to see all traffic within that network and inspect it for any threats. Suricata, in the IPS mode, will either `DROP` or `ALERT` depending on whether the rulesets detect any malicious intent. The basic logical flow for network traffic will be the following.

$$\texttt{Kali (Attacker)} \rightarrow \texttt{Ubuntu (Suricata IPS)} \rightarrow \texttt{Target System}$$

To achieve this lab environment for testing Suricata's IPS capabilities, we must first ensure that the VMs can communicate with one another. Using the logical flow for the network traffic model, we will configure the Ubuntu VM to act as the firewall to be able to intercept and route network traffic. We set the virtual network adapter for all virtual machines to VMnet 8. This will allow each VM to communicate with each other and the Internet as well. Additionally, to allow the Ubuntu VM to act as the firewall, we had to enable IP forwarding with the command `sudo sysctl -w net.ipv4.ip_forward=1`. Please see Table 1 to view the addressing for each virtual machine.

Table 1: Network Interface and Subnet Configuration for IPS Lab Environment

System	Interface	IP Address / Subnet
Ubuntu (Suricata IPS)	ens33	192.168.158.128/24
Kali (Attacker)	eth0	192.168.158.129/24
Windows (Target)	Ethernet	192.168.158.130/24

After the IPS lab environment was configured, we proceeded with testing Suricata's abilities for blocking common attack methods. We tested both a ping sweep attack using `PING` and a port scan using `NMAP`. After running both of these attacks, we noticed that Suricata did not alert or block the ping sweep attack. However, Suricata did in fact detect the `NMAP` port scan. After looking at the

logs of Suricata, we noticed that Suricata did not have the ruleset to be able to detect the ping sweep attack by default. After adding `drop icmp any any -> any any (msg:"ICMP Drop Test"; sid:1000001; rev:1;)` into Suricata's custom ruleset, Suricata blocked and alerted for both of the attacks. From the Kali VM's perspective, we can see that Suricata successfully blocked the attacks due to the commands not being able to run on the victim VM's IP address. In the logs of Suricata, the alert messages showed that both attacks were blocked with the message `ICMP Drop Test` and `NMAP SYN Scan Blocked`. These findings mean that Suricata, by default, did not have the necessary means to be able to block the ping sweep attack. This is likely due to the ping sweep attack not being categorized as a high-priority attack vector within the default ruleset. It is important to remember that with the correct rulesets in place, Suricata has immense capabilities.

H Future Work

There are also multiple ways this project could be expanded upon in the future. More attacks could be tested, and Suricata IPS mode could be evaluated. Implementing Suricata within a lab environment is just the tip of the iceberg. During the planning phase of this project, we were planning on integrating Suricata with Security Onion. This would give us a SIEM platform to receive visual alerts within one central platform. However, we decided that we did not have enough time to integrate Security Onion within our project.

I Lessons Learned

We learned a lot from this project, not only Suricata and how it works under the hood, but also indirect information as well. We learned new Kali Linux tools that we can use in our upcoming classes and our professional careers. We learned how attacks are classified using the MITRE ATT&CK framework and how some attacks are prioritized to be triaged first rather than others, depending on impact. We also learned valuable troubleshooting skills for Linux distributions. Choosing Suricata to conduct this research project was a great choice!