SCIENTIFIC CULTURE
www.sci-cult.com

SCIENTIFIC CULTURE, Vol. 12, No. 5.1, (2026), pp. 858-870
Open Access. Online & Print

DOI: 10.5281/zenodo.12511069

IMPROVING EFFECTIVENESS OF INTRUSION DETECTION SYSTEM BY INTEGRATING YARA AND SURICATA

Vandana Kadam [1]*, Rakesh Verma [2]

[1] *Research Scholar, Indian Institute of Management, Mumbai-400 087, India.*
[2] *Professor, Indian Institute of Management, Mumbai-400 0087, India.*

Received: 11/12/2025
Accepted: 25/02/2026

Corresponding author: Vandana Kadam
(Vandana.kadam.2015@iimmumbai.ac.in)

ABSTRACT

Intrusion detection system (IDS) are essential for safeguarding organizational resources; however, their practical deployment is often hindered by alert fatigue and the need for extensive rule configuration and tuning, which can reduce detection efficiency. YARA is a pattern-matching tool for detecting malware and intrusions indicators in cybersecurity contexts. This study investigates the integration of Suricata, an open-source IDS, with YARA to enhance intrusion detection performance. The implementation was conducted in a virtual environment and the integrated system was tested using live network traffic captured using Wireshark, a network monitoring tool. The results show that there is 20% reduction in the total number of alerts following the integration of Suricata and YARA rules, indicating a decrease in redundant or false alerts. These findings provide empirical evidence that IDS performance is strongly influenced by rule configuration and tuning strategies. The observed reduction in alerts confirms that YARA-based customization can improve IDS efficiency and mitigate alert-fatigue. The study highlights the potential of YARA–Suricata integration is an effective fine-tuning strategy and underscores its relevance for future research on automated rule optimization, false-positive reduction methods, and adaptive intrusion detection systems aimed at strengthening real-world cybersecurity operations.

KEYWORDS: Intrusion detection system (IDS); YARA; Suricata; intrusion detection; deployment of IDS; false alerts.

2.1. IDS and its types

Intrusion Detection Systems (IDS) are core components of cybersecurity infrastructure, designed to monitor network traffic or host activities for signs of malicious behavior or violations of security policies. IDS can be classified into two primary types Network based IDS (NIDS), which analyze network traffic, and host-based IDS (HIDS), which monitor activities on individual systems. IDS can also be categorized by detection method such as Signature based IDS and anomaly-based IDS [9].

Signature-based IDS rely on predefined patterns or signatures to identify known threats and are widely deployed due to their efficiency and interpretability. However, their effectiveness is limited when facing novel or evolving attacks, such as zero-day exploits. This limitation has driven the exploration of tools like YARA to enhance IDS capabilities [9]. Anomaly-based IDS aim to detect deviations from normal behavior, which may indicate potential threats offering improved detection of unknown threats but often suffering from higher false-positive rates. As a result, modern IDS deployments increasingly rely on Hybrid IDS which combines signature-based and anomaly-based approaches for more comprehensive detection.

2.2. YARA: Rule-Based Malware Detection Tool

YARA is an open-source rule-based pattern-matching tool developed by Victor Alvarez of VirusTotal and released on GitHub in 2013. It is widely used in malware research and cybersecurity operations to identify and classify malicious artifacts based on binary and textual patterns. YARA allows analysts to define custom rules that describe characteristics of malware families, enabling efficient detection across diverse datasets.

YARA rules consist of logical conditions that match strings, byte sequences, or structural attributes within files or memory. This rule-based approach supports precise malware classification and is effective in identifying threats such as ransomware, viruses, and worms. Owing to its platform-independent design, YARA can be deployed across heterogeneous environments, including Windows and Linux systems. Its flexibility and extensibility have led to its recognition as a versatile analytical tool within the cybersecurity community [10].

2.3. Application of YARA in Malware Detection and Forensic Analysis

A growing body of research has explored the application of YARA in malware detection and digital forensic investigations. Yildirim et al [10] demonstrated the use of YARA as an open-source tool for effectively detecting and removing the malware on a

1. INTRODUCTION

In the rapidly evolving landscape of cyber threats and attacks, protecting organizational systems from malware remains a critical challenge. Despite the widespread deployment of various security technologies such as anti-viruses, firewalls, intrusion detection and prevention system, malware continues to infiltrate systems and data through techniques including phishing, credential theft, and other threat actor strategies. This persistent threat environment underscores the need for more effective and adaptive security mechanisms.

Intrusion detection systems (IDSs) form critical components of information system security, playing a key role in ensuring system integrity. An IDS continuously monitors networks or systems to identify potential malicious activities or system policy violations and generates alerts that enable early response to security incidents. By providing timely warnings, IDSs help organizations mitigate the impact of cyberattacks before they escalate into significant operational or data losses [1–3].

However, traditional signature-based IDS face substantial limitations in the context of increasingly complex cyberthreats, including advanced malware and zero-day assaults like high false alarm rates. IDSs are prone to false alarms, organisations exert efforts in minimising both false positives and false negatives [4,5]. This tedious process of fine-tuning of IDS that requires expertise is carried out by admin team [6,7].

Reducing false alarms is a central objective in IDS and IPS(Intrusion Prevention System) fine-tuning, with an aim to maintain alert volumes within predefined organizational thresholds [8]. These limitations have driven the exploration of tools like YARA, a rule-based pattern-matching tool originally designed for malware detection to enhance the IDS capabilities promoting the use of hybrid models [9]

This study examines the integration of rule-based detection using YARA into an IDS framework and the challenges involved. It further evaluates how rule optimization can improve intrusion detection effectiveness, using Suricata as a case study to provide empirical insights into IDS fine-tuning.

This paper is organized as follows. Section presents the Theoretical Background, covers IDS and its types, YARA and its features, prior studies on YARA applications in cybersecurity and its effectiveness in IDS using case studies. Section 3 includes the methods and tools used, such as YARA, Suricata, and IDS. Section 4 presents the implementation and evaluation of the YARA–Suricata integration. Section 5 is contains the results, discussions, limitation and future scope.

2. THEORETICAL BACKGROUND

has supported its adoption in malware detection, digital forensics, incident response, and cyber threat intelligence.

Yildirim et al. [10] applied YARA, in digital forensics and incident response to identify malware on compromised systems. Their study detailed the rule creation and demonstrated YARA's adaptability and and effectiveness across Windows and Linux environments. Mahdi and Trabelsi [11] further demonstrated that carefully designed rule set can improve the detection accuracy and efficiency by introducing The static malware detection approach is implemented by analyzing various file characteristics and proposed system with enhanced operational efficiency and scalability supporting the parallel scanning of multiple files.

Alam et al. [17] investigated the integration of YARA-based scanning into a Raspberry Pi platform for detecting the malware and ransomware introduced through USB devices using the byte and string pattern matching and suggested that this approach is scalable and suitable for real world endpoint security applications and suggested that combining static and dynamic analysis could further strengthen detection capabilities. Naik et al. [12] evaluated automatically generated YARA rules using yabin yaraGenerator and yarGen targeting a ransomware families like Locky, Cryptowall and WannaCry, reporting improved detection performance through the incorporation of fuzzy hashing.

Lockett [14] compared rule based YARA detection with cryptographic and fuzzy hashing approaches and demonstrated that YARA rules crafted using binary patterns are more effective in detecting the modified or obfuscated malware. Altarawani et al. [13] targeted Emotet malware, a highly polymorphic and evasive threat, by using YARA. They used python to develop custom rules by utilizing behavioral and memory analysis focussing on metadata, file size, network activity, suspicious behavior. Their study suggested that integrating YARA with machine learning techniques could further improve adaptability to evolving threats.

Costin and Zaddach [18] addressed IoT malware analysis by proposing an open source framework for systematic and reproducible research. They highlighted the Internet of Things (IoT) threat environments and the need for structured analytical frameworks, reinforcing the importance of signature-based and rule-driven tools in emerging domains.

Similarly, Khalid et al. [15] proposed an automated framework for generating high-quality YARA rules with minimal expert involvement. The framework employs a Naives Bayes classifier to score and select optimal string signatures, enabling automated rule compromised system and created a defence mechanism by using YARA in forensic investigation. Their study evaluated multiple attack scenarios, including phishing attacks, and validated Yara's effectiveness across Windows and Linux system. The authors have inspected the attacker's techniques and persistence mechanism used in cyberattacks on organizations systems using YARA model, an open source tool and confirmed the effectiveness in of YARA in malware detection.

Several authors have used YARA for static malware detection. Mahdi and Trabelsi [11] investigated the use of YARA in static malware analysis, while Naik et al. [12] proposed automated approaches for generating YARA rules to improve scalability and detection coverage. Altarawani et al. [13] focused on YARA-based detection of specific malware types, such as Emotet, further demonstrating its applicability in targeted threat identification. Collectively, these studies highlight YARA's effectiveness as a malware detection tool across diverse cybersecurity contexts.

2.4. Related Works

Several studies have been conducted on the use of YARA for detection of mark of intrusions. Naik et al. [12] studied the performance of automatically generated YARA rules using three open-source tools such as yarGen, yaraGenerator, and yabin, and proposed the use of fuzzy hashing to enhance effectiveness. Adam Lockett [14] evaluated the effectiveness of YARA rules for malware detection and classification, comparing it to cryptographic and fuzzy hashing approaches. Altarawani et al. [13] introduced a YARA rule-based detection tool for identifying and mitigating Emotet malware, a sophisticated and polymorphic threat that evades traditional detection methods. Meanwhile, Khalid et al. [15] introduced an automated framework for generating high-quality YARA rules for malware detection, aiming to minimize expert intervention. Furthermore, Mahdi and Trabelsi [11] applied YARA for static files malware detection and Kovalchuk [16] studied the evolution of malware from early viruses to modern cyber threats. As the complexity increases, the need for deep cybersecurity knowledge and a balance between detection accuracy and rule complexity is essential.

2.4.1. Existing Studies on YARA's Application in Cybersecurity

Recent research highlights YARA as an effective signature-based and pattern-matching tool for detecting and mitigating a wide range of cyber threats, including phishing, ransomware, distributed denial-of-service (DDoS) attacks, and polymorphic malware. Its ability to define custom rules using strings, binary patterns, metadata, and behavioral indicators

and Tactics, Techniques, and Procedures (TTPs) in proactive defense, and underscoring the relevance of rule-based tools like YARA for applying technical CTI for real-time threat detection and response. Prajapati et al. [20] examined ransomware detection and forensic analysis using Windows-based tools, including YARA and confirmed its effectiveness in forensic investigations across major ransomware families. Table 1 summarizes recent studies on the application of YARA in cybersecurity, highlighting their focus areas, methodologies, and key contributions.

Table 1: Recent Studies on application of YARA in cyber security.

Author	Year	Focus Area	Methodology /tools	Key Contributions
Costin & Zaddach	2020	IoT Malware Analysis	IoT Sand box and archival	Create reproducible Framework for IoT Malware
Adam Lockett	2021	Rule based vs. hash based Malware	Cryptographic hash vs. YARA rules.	Showed YARA's flexibility over static hashes
Yildirim et al. [10]	2023	YARA for cyber attack detection (Phishing and DDOS)	Custom YARA rules, static analysis	Demonstrated YARA rule creation and use in DFIR
Alam et al. [17]	2023	USB-based malware/ransomware detection	Raspberry Pi, Tkinter GUI, Aho-Corasick, YARA	Developed portable real-time USB malware scanner
Altarawani et al.	2023	Emotet detection	Custom YARA rules, behavioral indicators, Python	Designed Emotet-specific detection tool
Khalid et al. [15]	2023	Automated YARA rule framework	Naïve Bayes classifier, 5-module system	Automated high-quality YARA rule generation
Mahdi & Trabelsi [11]	2024	Static malware detection using YARA	7 custom YARA rules, file analysis	Developed static detection framework using custom rules
Kovalchuk	2024	Malware evolution and defenses	Historical, empirical malware analysis	Outlined malware evolution and defense enhancements
Prajapati et al. [20]	2024	Ransomware forensic analysis	YARA, FTK Imager, Volatility	Examined ransomware families using forensic tools
Naik et al.	2025	Automated YARA rule generation	yarGen, yaraGenerator, yabin, SSDEEP	Improved detection accuracy with fuzzy hashing
Gyebnár & Magyar [19]	2025	CTI integration with YARA	Operationalization of IoCs and TTPs	Enhanced detection via CTI indicators

demonstrated the effectiveness of YARA in detecting malicious PHP (Hypertext Preprocessor) files and webshells. Hybrid approaches combining YARA rules with Convolutional Neural Networks (CNN) have achieved 99.02% accuracy, demonstrating YARA's potential in identifying web-based threats. In android malware detection the hybrid models using deep learning (CNNs) and feature engineering have achieved high accuracy, with some models reporting up to 99.6% accuracy [21-24]. While these studies primarily focus on CNNs, several other works suggest that combining rule-based approaches like YARA with machine learning can further improve detection rates and adaptability to new generation. Experimental results showed high precision and strong detection performance for both known and previously unseen malware, outperforming existing YARA rule-generation tools.

Beyond rule-based detection, broader malware evolution and threat intelligence contexts have been studied. Kovalchuk [16] analyzed the progression of malware from early viruses to modern threats such as ransomware and botnets, emphasizing the need for continuously evolving defense mechanisms. Gyebnár and Magyar [19] focused on the operationalization of Cyber Threat Intelligence (CTI), emphasizing the role of Indicators of Compromise (IoCs)

2.5. Case Studies on YARA's Effectiveness in IDS Contexts

Recent case studies have examined YARA's effectiveness in IDS and security contexts, including web application security (Yildirim et al., 2023), android malware protection [21-24], Windows environment, and IoT and other malware security [25,26]. These studies have enhanced YARA rules using fuzzy hashing, improving detection outcomes without increasing complexity [12,27,28]. The implementation of FPGA-based hardware accelerator for YARA is used by Singapura et al. [29].

In web application security, studies have

throughput improvements of 8.8× to 14.5× (12.85 Gbps to 21.8 Gbps) compared with software-only implementation's 1.45 Gbps, making it suitable for high-speed networks.

Overall, these case studies illustrate YARA's versatility and effectiveness across diverse IDS contexts, from web security to high-performance network monitoring.

3. PROPOSED RESEARCH METHODOLOGY

The objective of this study is to design YARA rules for detecting selected file types and patterns and to evaluate the impact of integrating these rules into Suricata, an IDS. The study examines changes in the number of alerts or false alarms generated before and after YARA integration, with particular attention to alert reduction and false positives.

3.1. Method and Tools

The experimental setup was implemented on a 64-bit x64-based processor running Windows 10 operating system, with an Ubuntu 24.04 virtual machine used for IDS deployment. The primary software tools used in this study include are YARA64 for rule creation, Suricata as an IDS, and Wireshark for network traffic capture. The methodology followed a sequential process, as illustrated in Figure 1.

1. YARA rules were crafted to address predefined threat conditions and tested using sample data to validate rule functionality.

threats [25,30].

For Windows malware detection, CNN-based detectors using behavioral and image-based features have shown strong performance [31-33]. Although direct integration with YARA is less frequently reported, the literature highlights the potential of combining static (YARA) and dynamic (CNN) analysis for robust detection [25,26,30]. In IoT and other malware, YARA rules have been successfully used to detect diverse malware families with studies emphasizing the importance of regularly updated rule sets and hybrid detection strategies to reduce false positives [25,26].

Research on fuzzy hashing has shown that enhanced YARA rules incorporating similarity-based matching outperform traditional YARA rules across malware and goodware datasets. Experimental results on ransomware datasets, including WannaCry, Locky, Cerber, and CryptoWall, indicate improvements in similarity detection rates by 4.9%–11.3% over standard YARA rules, depending on the dataset and method used [27,28]. Adding fuzzy hashing does not significantly increase rule complexity or system overhead, rather complements YARA's string/pattern matching by identifying structural similarities in files, which is useful for detecting obfuscated or slightly modified malware [12,27,28].

Hardware acceleration has also been to explored address performance constraints in high-speed environments Singapura et al. [29] proposed a FPGA-based hardware accelerator for YARA, achieving

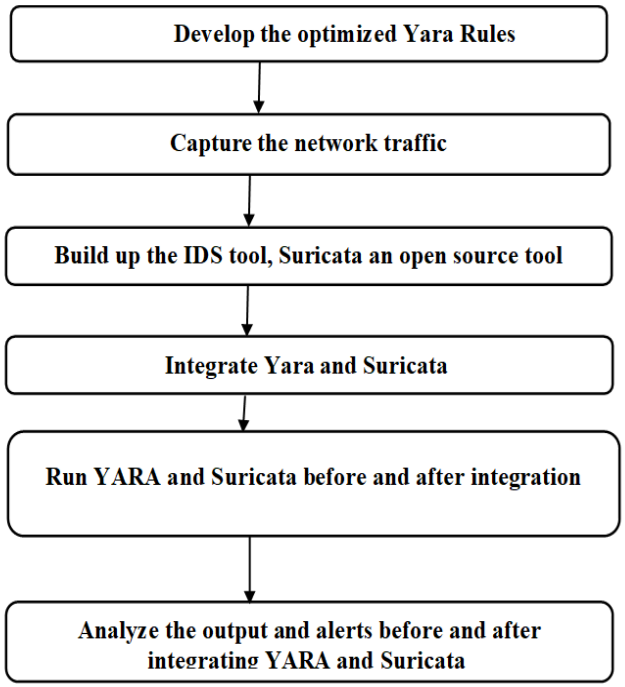


Figure 1: Proposed Methodology

2) Tracking Malware families: It is used for tracking malware families wherein rules are crafted to identify and track malware families. This is generally, used for memory tracking and requires unpacked samples (or memory) and has a low tolerance for false positives. *3) Hunting rule:* The purpose is to identify generic malware characteristics and hunt rapidly developing similar malware. This category has high false positive and is not useful for alerting and blocking.

Signature or strings of malware are compared with signature or strings of known malware in YARA rules for identifying the malware. These rules include signatures or strings related to known malware that is tried to match the targeted files, directories, or processes [11].

4. INTEGRATING YARA WITH SURICATA

YARA enhances IDS by providing a robust and powerful framework for pattern matching, enabling the detection of known malware signatures. It is one of the important tool for enhancing Intrusions Detection Systems (IDS), covering both Network-based (NIDS) and Host-based (HIDS) placements. Its integration can be applied in both HIDS and NIDS.

4.1. How YARA Integrates with IDS

YARA rules are highly effective for identifying known malware signatures by matching specific patterns, strings, or byte sequences within files or network traffic. This capability is central to both HIDS monitoring endpoints and NIDS, monitoring network traffic thus used for pattern matching and signature detection. [11,12].

YARA allows analysts to craft custom rules tailored to specific threats or environments, increasing detection accuracy and adaptability to new malware variants providing customizability and flexibility [12,25].

Automated YARA rule generation and integration with machine learning can further enhance detection rates and reduce analyst workload, making IDS more responsive to evolving threats which adds efficiency [28,30,(Naik et al., 2020b; Raff et al., 2020; Si et al., 2022).

YARA rules can scan files on a system for malicious patterns, such as specific API calls or file structures. For example, signatures derived from API sequences can be implemented into YARA rules to improve HIDS [30] YARA can inspect network traffic for patterns indicative of malware, such as command-and-control communications or malicious payloads.

2. Network traffic was then captured using Wireshark to stimulate predefined network scenarios. The captured traffic will be used for evaluation.
3. Suricata was executed on the captured network traffic and the alerts were measured. The output was recorded using eve.json, stats.log and fast.log files.
4. Subsequently, YARA rules were integrated into Suricata through configuration changes, after which Suricata was re-executed on the same traffic and the resulting alerts were analyzed for comparison.

3.1. Method and Tools

YARA rule writing method: It works in two-step such as 1) Search all patterns listed in the rules regardless where they are in the file. 2) Evaluate the condition in each rule.

YARA rules consist of three parts: metadata, strings, and conditions. Strings may be defines as text, hexadecimal, or regular expressions.

There are several use cases for deploying YARA, which has contributed to its immense popularity within the information security field [12] YARA rules can recognize and categorize malware, Identify fresh samples according to family specific patterns. The proactive implementation of custom-written or bespoke YARA rules may strengthen the security of an organization and rules can be used to detect samples and compromised devices [11].

3.2. Suricata

Suricata is an open-source network security engine capable of functioning as a Network Security Monitoring (NSM) tool, Intrusion Prevention System (IPS), or IDS. Created by the Open Information Security Foundation (OISF), Suricata supports real-time packet inspection, protocol analysis, and file extraction from network traffic. Its multi-threaded architecture enables efficient processing of high throughput network data.

The EVE.JSON output, which enables smooth integration with Security Information and Event Management (SIEM) tools like ELK Stack (Elasticsearch, Logstash, Kibana), is one of Suricata's potent advantages.

3.3. Categorization of YARA rule:

Generally, YARA rules can be categorized based on purposes are as follows:

1) Malware identification: wherein YARA rules are crafted to identify malware on the disk. This is usually used for early delivery stage identification (scripts, docs, etc) but has a short life span and low tolerance for false positives.

Table 2: YARA's role in Host-based and Network -based IDS

IDS Type	YARA Application	Benefits	Citations
HIDS	Scans files, processes, and memory on endpoints using YARA rules	Detects known malware, rootkits, and suspicious activity	(Naik et al., 2020; Mahdi & Trabelsi, 2022) [12,
NIDS	Analyzes network packets and payloads for malicious patterns	Identifies malware, exploits, and command and control traffic	(Naik et al., 2020; Mahdi & Trabelsi, 2022) [12,

Step 2: Writing YARA rules for hunting threats
Step 3: Testing the YARA rules on Test Sample.
Step 4: Building up or Installing Suricata open source IDS tool.
Step 5: Integrating YARA and Suricata
Step 6: Measuring the number of alerts before after integrating Yara and Suricata.

4.3.1. Building up Suricata on ubuntu virtual machine

To build up Suricata for intrusion detection first set-up Suricata by installing it, then verify does it is installed properly. Once verified to start Suricata enable the Suricata service, start its service and check its status. Following are the commands used:

To install Suricata

sudo apt install -y suricata

To verify Suricata is installed properly

suricata -v

To start Suricata enable Suricata service, start it, then check the status as follows

sudo systemctl enable suricata
sudo systemctl start suricata
sudo systemctl status suricata

Integrating YARA and Suricata

The prerequisite for integration and to bring clarity in work are: i. Save all the rules at one location such as, /home/vandana/suryara (a path to folder where all .yar files are saved) ii.Save all the test data in a folder, named suryara_traffic on any given path such as /home/vandana/suryara_traffic just to bring convenience. And all output files generated will be saved in created folder named, suryara_output on this given path such as /var/log/Suricata/suryara_output

Configure the suricata.yaml file as follows:

The below screenshots show rules in detect_pdf_png.yar will be scanned.

Coscia et al., (2025) has given a notable example of YARA's integration is the APIARY tool, which automates the generation of YARA rules based on API calls. APIARY improves detection rates by identifying distinctive APIs that differentiate malware from benign software, achieving high effectiveness without relying on additional data like network connections. Additionally, projects like YAIDS (Yara as an Intrusion Detection System) demonstrate practical implementations of YARA in real-time network monitoring.

Integrating YARA with Suricata typically involves configuring Suricata to call YARA for rule-based scanning during its operation. By default, Suricata processes network traffic, often in the form of .pcap files or live network traffic. However, if you want to test with a .txt file instead of .pcap files

4.2. How to Use YARA

How to run YARA rule:
$yara your_rule.yar your_file.txt
Sample YARA rule is as shown below

```
rule Example_One
{
strings:
$string1 = "pay immediately"
$Hex_PDF = { 25 50 44 46 }
$Hex_PNG = { 89 50 4E 47 0D 0A 1A 0A }
$MaliciousWeb1 = "www.scamwebsite.com"
$MaliciousWeb2 = "www.notrealwebsite.com"
$Maliciousweb3 = "www.freemoney.com"
$AttackerName1 = "hackx1203"
$AttackerName2 = "Hackor"
$AttackerName3 = "Hax"
condition:
any of them
```

4.3. Implementation and Testing

Following are the steps followed for implementing YARA rules on Suricata, an IDS tool

Implementation Details:
Step 1: Installing Ubuntu, YARA

```
detection:
  enabled: yes
  yara:
    enabled: yes
    files:
      - /home/vandana/suryara/detect_pdf_png.yar
```

Figure 2: Shows the location of file where rules are written in .yar file

The Output files will be created in the path shown below screenshot:

```
    dp: 443


  yara:
    enabled: yes
    rules: /var/log/suricata/suryara_output
```

Figure 3: Shows the location of file where output is obtained.

Case 1: Detection of only .pdfs and .png files in a specified location

The detection is conducted in two stages with by enabling Yara rules in Suricata and without enabling the Yara rules in Suricata testing in two stages:

4.3.3 Stage 1: Without YARA Rule0073

The output will be saved in folder suryara_output shown on the following path: /var/log/suricata/suryara_output as eve.json, fast.log, stats.log files.

4.3.2. Running Suricata without and with YARA rules:

```
vandana@vandana-VirtualBox:~/suryara$ ls
detection.yar            emotet_malware.yar  stats.log
detectmoneydemand.yar    eve.json            suricata.log
detect_pdf_png.yar       fast.log            Task1_250328_215644.pdf
detect_urls_regExp.yar   preventwebsite.yar  test_rule.yar
vandana@vandana-VirtualBox:~/suryara$ yara detect_pdf_png.yar /home/vandana/sury
ara
Example_Three /home/vandana/suryara/Task1_250328_215644.pdf
vandana@vandana-VirtualBox:~/suryara$
```

Figure 4: Detected .pdf file in specified location

```
[11412 - Suricata-Main] 2025-04-09 15:20:11 Info: suricata: Setting engine mode
to IDS mode by default
[11412 - Suricata-Main] 2025-04-09 15:20:11 Info: exception-policy: master exce
tion-policy set to: auto
[11412 - Suricata-Main] 2025-04-09 15:20:11 Warning: ioctl: Failure when trying
to get MTU via ioctl for 'eth0': No such device (19)
[11415 - Suricata-Main] 2025-04-09 15:20:11 Info: logopenfile: fast output devi
e (regular) initialized: fast.log
[11415 - Suricata-Main] 2025-04-09 15:20:11 Info: logopenfile: eve-log output d
vice (regular) initialized: eve.json
[11415 - Suricata-Main] 2025-04-09 15:20:11 Info: logopenfile: stats output dev
ce (regular) initialized: stats.log

^C
vandana@vandana-VirtualBox:~/suryara$  'select(.event_type=="alert")' /var/log/
uricata/eve.json | wc -l
select(.event_type=="alert"): command not found
0
vandana@vandana-VirtualBox:~/suryara$ jq 'select(.event_type=="alert")' /var/lo
/suricata/eve.json | wc -l
4584
```

Figure 5: Output The number of threats detected by Suricata is 4584

```
01:40:12 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
01:50:05 PM     all     59.91      0.34      2.44      0.13      0.00     37.19
02:00:04 PM     all     52.54      0.00      1.22      0.01      0.00     46.22
02:10:04 PM     all     56.13      0.00      1.55      0.02      0.00     42.30
02:20:15 PM     all     65.07      0.01      2.23      0.04      0.00     32.65
02:30:14 PM     all     56.28      0.00      1.56      0.03      0.00     42.13
02:40:05 PM     all     59.67      0.05      1.84      0.05      0.00     38.38
02:50:04 PM     all     63.19      0.01      2.13      0.04      0.00     34.63
03:00:10 PM     all     63.78      0.02      4.11      0.09      0.00     32.01
Average:        all     59.59      0.05      2.14      0.05      0.00     38.17
vandana@vandana-VirtualBox:~/suryara$
```

Figure 6: Measuring the systems performance metrics such as throughput and resource usage

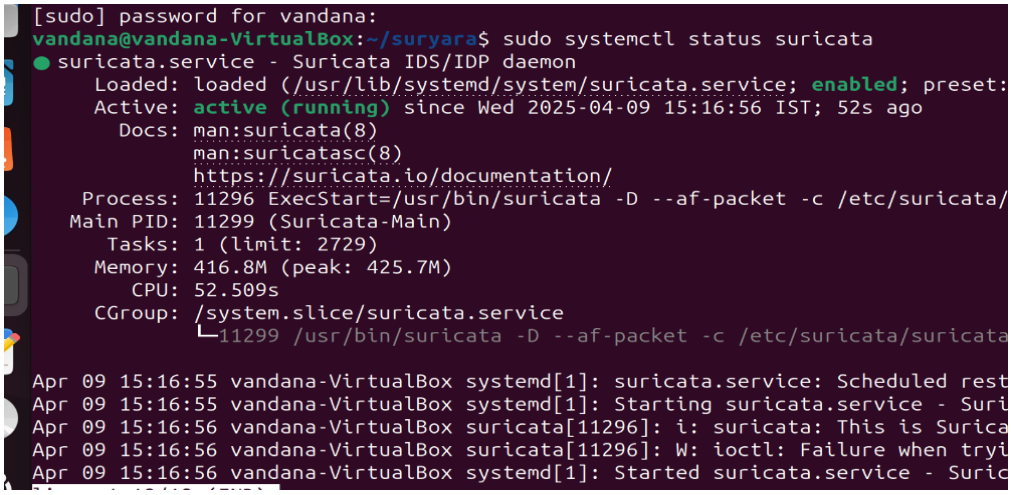


Figure 7: View logs

Integrate the YARA rules into the IDS.

4.3.4. Stage 2: With YARA Rules

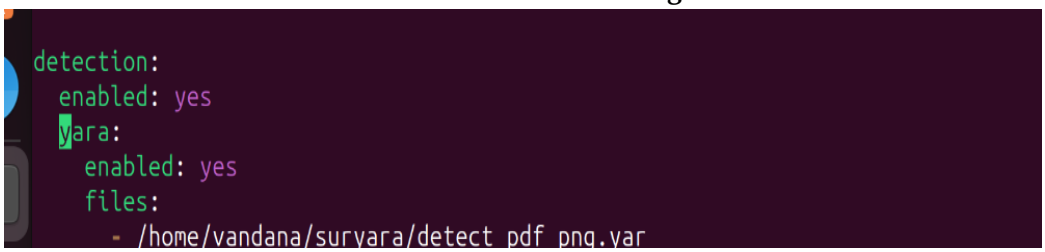
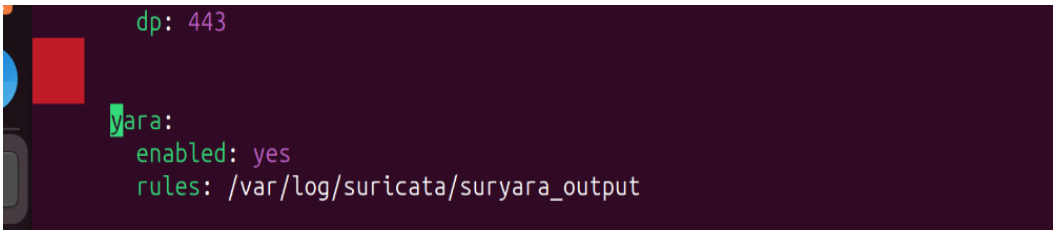


Figure 8: Showing the enabling the integration of YARA on Suricata.

Below screen shows configuration in which the path is created where output will be saved such as

Configuring the path in which output will be saved:



eve.json , stats.log and fast.log files will be created.

Figure 9. Showing the Configuration in which the path is created where output will be saved.

Rerun the IDS on the same dataset or traffic simulation. Record the same metrics as in Stage 1.

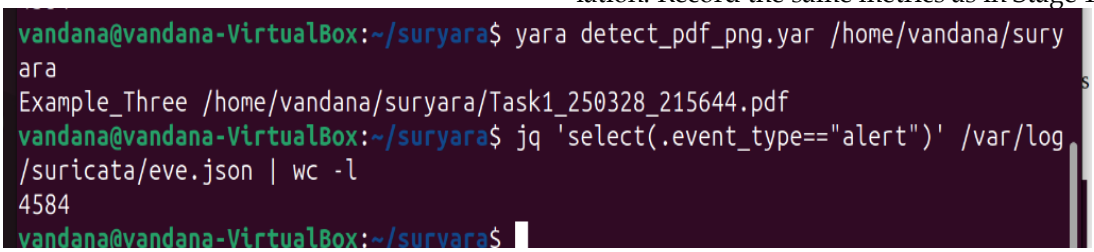


Figure 10: Output Record: Number of threats detected by Suricata with yara.=4584 same as previous.

```
01:40:12 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
01:50:05 PM     all     59.91      0.34      2.44      0.13      0.00     37.19
02:00:04 PM     all     52.54      0.00      1.22      0.01      0.00     46.22
02:10:04 PM     all     56.13      0.00      1.55      0.02      0.00     42.30
02:20:15 PM     all     65.07      0.01      2.23      0.04      0.00     32.65
02:30:14 PM     all     56.28      0.00      1.56      0.03      0.00     42.13
02:40:05 PM     all     59.67      0.05      1.84      0.05      0.00     38.38
02:50:04 PM     all     63.19      0.01      2.13      0.04      0.00     34.63
03:00:10 PM     all     63.78      0.02      4.11      0.09      0.00     32.01
03:10:07 PM     all     67.49      0.02      5.83      0.63      0.00     26.02
03:20:11 PM     all     66.45      0.01      4.22      0.31      0.00     29.00
03:30:04 PM     all     59.46      0.03      3.25      0.51      0.00     36.75
Average:        all     60.92      0.04      2.76      0.17      0.00     36.11
vandana@vandana-VirtualBox:~/suryara$
```

Figure 11: Measuring the systems performance metrics such as throughput and resource usage.

Now Changing the configuration in suricata.yaml file, It will scan all the .yar rules in suryara folder /home/vandana/ suryara which is shown in Figure 12.

Case 2: Detection of alerts on live traffic captured using wireshark tool

```
#   - include1.yaml
#   - include2.yaml

detection:
  enabled: yes
  yara:
    enabled: yes
    files:
      - /home/vandana/suryara
```

Figure 12: Location of .yar files in suryara folder for Case 2

Running Suricata again on live traffic captured as .pcap files and **showing the alerts without enabling YARA rules is shown in Figure 13.**

```
vandana@vandana-VirtualBox:~$ sudo suricata -r /home/vandana/suryara_traffic/tra
fficcaptured.pcap -l /var/log/suricata/suryara_output
i: suricata: This is Suricata version 7.0.3 RELEASE running in USER mode
i: threads: Threads created -> RX: 1 W: 2 FM: 1 FR: 1   Engine started.
i: suricata: Signal Received.  Stopping engine.
W: pcap: 1/2th of packets have an invalid checksum, consider setting pcap-file.c
hecksum-checks variable to no or use '-k none' option on command line.
i: pcap: read 1 file, 272 packets, 52095 bytes
vandana@vandana-VirtualBox:~$ jq 'select(.event_type=="alert")' /var/log/suricat
a/suryara_output/eve.json | wc -l
22920
vandana@vandana-VirtualBox:~$
```

Figure 13: Output: Number of Alert without enabling YARA rules: 22920 on live traffic.

Running Suricata again on live traffic captured with wireshark as trafficcaptured.pcap file and showing the alerts after integrating YARA is shown in Figure14.

```
vandana@vandana-VirtualBox:~$ sudo suricata -r /home/vandana/suryara_traffic/tra
fficcaptured.pcap -l /var/log/suricata/suryara_output
[sudo] password for vandana:
i: suricata: This is Suricata version 7.0.3 RELEASE running in USER mode
i: threads: Threads created -> RX: 1 W: 2 FM: 1 FR: 1   Engine started.
i: suricata: Signal Received.  Stopping engine.
W: pcap: 1/2th of packets have an invalid checksum, consider setting pcap-file.c
hecksum-checks variable to no or use '-k none' option on command line.
i: pcap: read 1 file, 272 packets, 52095 bytes
vandana@vandana-VirtualBox:~$ jq 'select(.event_type=="alert")' /var/log/suricat
a/suryara_output/eve.json | wc -l
18336
```

Figure 14: Alerts generated with enabling YARA rules = 18336 on live traffic captured.

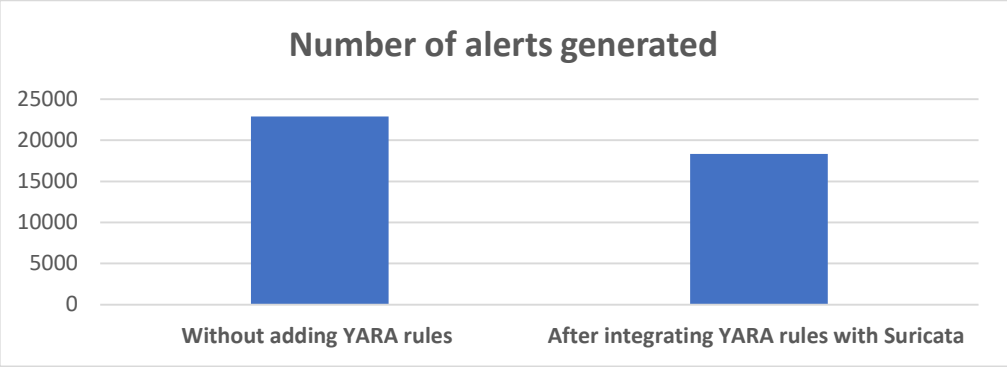


Figure 15: Number of Alerts generated before and after integrating YARA rules with Suricata on live traffic captured.

positives generated by traditional signature-based systems. Undue higher alerts can distress security analysts and reduce the applied effectiveness of an IDS.

By integrating YARA rules, Suricata gains the ability to differentiate between benign traffic patterns and genuinely malicious content. This selective detection approach reduces false positives while maintaining, or even improving, detection accuracy. The graph therefore validates the role of YARA as an optimization layer instead of just an alert-generating extension.

From a research perspective, this result supports hybrid IDS architectures that prioritize alert precision over alert volume. The integration shows more efficient and manageable security monitoring.

5.2. Limitation and Future Scope

There are certain limitations of this study, first the evaluation is based on specific set of YARA rules and live traffic captured for limited period of observation. The reduction in alert is observed, which may vary with different traffic profiles, rule sets or attack scenarios when applied.

Second limitation is, here focus is only on alert count other metrics such as detection accuracy, true positive rate, false positive rate, or precision are uncovered.

Finally, the study is dependent on manually crafted rules instead of automated rule generation, thereby limiting the generalization across diverse threat landscape.

5. CONCLUSION

5.1. Results and Discussions

YARA is a powerful tool for enhancing Intrusion Detection System, offering precise and automated malware detection through pattern matching.

Figure 15 presents a result obtained, comparison of the number of alerts generated by the Suricata Intrusion Detection System (IDS) before and after the integration of YARA rules on live network traffic. Prior to YARA integration, Suricata depends mainly on default signature based and protocol level rules. During this phase, the IDS generates a higher number of alerts, many of which are triggered by generic patterns, protocol anomalies, or benign traffic behaviours. Before integration of Suricata and YARA, the number of alerts generated by IDS are 22920 and after integration are 18336 showing the decrease of 20% in alerts generated which clearly supports reduction in false or redundant alerts.

This reduction is not indicating weak detection; rather, it reflects improved alert filtering and rule precision. And also, YARA rules enable content-aware inspection and precise pattern matching within files and payloads and enhances the IDS through customizable and precise detection mechanisms.

It is observed that the decrease in alert volume after YARA integration is a significant indicator of successful IDS fine-tuning. One of the major challenges in intrusion detection research is the high rate of false

Vandana Kadam: Conceptualization, Investigation, implementation, Writing- original draft, Writing review and editing.

Rakesh Verma: Supervision, Conceptualization, Analysis and Interpretation of results.

Data Availability Statement: The data used in the study will be available on request

Acknowledgments: I wholeheartedly thank my research guide Dr. Rakesh Verma for his constant support.

Conflicts of Interest: The authors declare no conflict of interest associated with this research work**.**

ABBREVIATIONS

The following abbreviations are used in this manuscript:

From the finding of this work it is evident the performance of IDS is highly dependent on rule configuration and tuning. The reduction in alerts count justifies that the YARA based customization can improve the IDS efficiency thereby showing the significance and future scope for automated rule optimization, false positive reduction techniques and adaptive intrusion detection system

In General, the graph confirms that YARA–Suricata integration is an effective fine-tuning strategy that enhances IDS efficiency, reduces alert fatigue, and strengthens real-world cybersecurity operations.

In General, this work confirms that Suricata-YARA integration is an effective fine-tuning strategy that enhances IDS efficiency, reduces alert fatigue, and strengthens real-world cybersecurity operations.

AUTHOR CONTRIBUTIONS

ID	Intrusion Detection
IDS	Intrusion Detection system
YARA	Yet Another Recursive Acronym

REFERENCES

1. Heidari, A., & Jabraeil Jamali, M. A. (2023). Internet of Things intrusion detection systems: a comprehensive review and future directions. *Cluster Computing*, *26*(6), 3753-3780. https://doi.org/10.1007/s10586-022-03776-z
2. Patel, A., Qassim, Q., & Wills, C. (2010). A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, *18*(4), 277-290. https://doi.org/10.1108/09685221011079199
3. Kadam and Verma et al., 2024
4. Hubballi, N., & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. Computer Communications, 49, 1-17. https://doi.org/10.1016/j.comcom.2014.04.012
5. Maggi, F., Matteucci, M., & Zanero, S. (2009). Reducing false positives in anomaly detectors through fuzzy alert aggregation. *Information Fusion*, *10*(4), 300-311. https://doi.org/10.1016/j.inffus.2008.07.006
6. Debar & Viinikka, 2004
7. Diaz-Verdejo et al., 2022
8. Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of network and computer applications*, *36*(1), 16-24.
9. Abulayhi et al., 2021
10. Yıldırım, K., Demir, M. E., Keleş, T., Yıldız, A. M., Doğan, Ş., & Tuncer, T. (2023). A YARA-based approach for detecting cybersecurity attack types. Firat University Journal of Experimental and Computational Engineering, 2(2), 55–68. https://doi.org/10.5505/fujece.2023.09709
11. Mahdi, R. H., & Trabelsi, H. (2022). Role of YARA tool in intrusion detection systems: A review. In Proceedings of the International Conference on Computer Systems and Technologies (pp. 185–190). IEEE. https://doi.org/10.1109/CSCTIT56299.2022.10145723.
12. Naik, N., Jenkins, P., Cooke, R., Gillett, J., & Jin, Y. (2020a). Evaluating automatically generated YARA rules and enhancing their effectiveness. In Proceedings of the IEEE Symposium Series on Computational Intelligence (pp. 1146–1153). https://doi.org/10.1109/SSCI47803.2020.9308179
13. Altarawani et al., 2023
14. Lockett, A. (2021). Assessing the effectiveness of YARA rules for signature-based malware detection and classification (arXiv Preprint No. 2111.13910). https://doi.org/10.48550/arXiv.2111.13910

15. Khalid, M., Ismail, M., Hussain, M., & Durad, M. H. (2020). Automatic YARA rule generation. In Proceedings of the International Conference on Cyber Warfare and Security. https://doi.org/10.1109/ICCWS48432.2020.9292390
16. Kovalchuk, D. (2024). Malware development: From early viruses to modern cyber threats. Visnyk Cherkaskoho Derzhavnoho Tekhnolohichnoho Universytetu, 29(3), 10–20. https://doi.org/10.62660/bcstu/3.2024.10
17. Alam, M. N., Singh, A., Kumari, M., Agrawal, P., Dubey, P., & Kumar, A. (2023). Detection and prevention of malware and ransomware threats using malicious string analysis. In Proceedings of the 2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology (ICSEIET) (pp. 93–98). IEEE. https://doi.org/10.1109/ICSEIET
18. Costin and Zaddach (2020)
19. Gyebnár, G., & Magyar, S. (2025). Consumption of technical threat intelligence indicators. In Proceedings of the IEEE Symposium on Applied Machine Intelligence and Informatics (pp. 177–182). https://doi.org/10.1109/SAMI63904.2025.10883294
20. Prajapati et al.
21. Wang, W., Zhao, M., & Wang, J. (2019). Effective Android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, 10, 3035–3043. https://doi.org/10.1007/s12652-018-0803-6
22. Yadav et al., 2022
23. Dong, S., Shu, L., & Nie, S. (2024). Android malware detection method based on CNN and DNN hybrid mechanism. IEEE Transactions on Industrial Informatics, 20, 7744–7753. https://doi.org/10.1109/TII.2024.3363016
24. Wasif et al., 2025
25. Patil, V., N. M., P. M., & Singh, A. (2025). Effectively writing YARA rules to detect malware. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/IJRASET.2025.66535
26. Alsattam, F., Al-Akhras, M., Almasri, M., & Alawairdhi, M. (2020). Rule-based approach to detect IoT malicious files. Journal of Computer Science, 16, 1203–1211. https://doi.org/10.3844/jcssp.2020.1203.1211
27. Naik et al., 2019
28. Naik, N., Jenkins, P., Savage, N., Yang, L., Boongoen, T., & Iam-On, N. (2020b). Embedded YARA rules: Strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis. Complex & Intelligent Systems, 7, 687–702. https://doi.org/10.1007/s40747-020-00233-5
29. Singapura et al., 2016
30. Si, Q., Xu, H., Tong, Y., Zhou, Y., Liang, J., Cui, L., & Hao, Z. (2022). Malware detection using automated generation of YARA rules on dynamic features. In Lecture Notes in Computer Science (pp. 315–330). Springer. https://doi.org/10.1007/978-3-031-17551-0_21
31. Vasan, D., Alazab, M., Wassan, S., Safaei, B., & Qin, Z. (2020). Image-Based malware classification using ensemble of CNN architectures (IMCEC). Comput. Secur., 92, 101748. https://doi.org/10.1016/j.cose.2020.101748
32. Kumar, S., & Panda, K. (2023). SDIF-CNN: Stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification. Applied Soft Computing, 146, Article 110676. https://doi.org/10.1016/j.asoc.2023.110676
33. Feroz, R., Aslam, M., Fuzail, M., Aslam, N., & Abid, M. (2025). Hybrid deep learning effectiveness of image-based malware detection. Kashf Journal of Multidisciplinary Research. https://doi.org/10.71146/kjmr415