Enhancing Cybersecurity Resilience with CYBRANA: A Cyber YARA/YAML-Based Resilience Firewall Solution Applied with Next-Gen AI

Aaditya Rengarajan
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
aadityarenga@gmail.com

Dr. G. R. Karpagam
Computer Science and Engineering
PSG College of Technology
Coimbatore, India
grk.cse@psgtech.ac.in

2024 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI) | 979-8-3503-7687-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/CVMI61877.2024.10782249

***Abstract*—The ever-increasing volume of server requests puts digital infrastructure at more risk of cyberattacks. This paper introduces CYBRANA, a cyberattack detection and mitigation system powered by AI. CYBRANA uses a Random Forest model to look into firewall and server logs for potential malicious threats. After analysis of flagged logs to extract request paths, CYBRANA then maps YAML rules to look for known attack pattern detection. Upon successful detection, CYBRANA classifies the attack type and severity (using CVSS scoring) by mapping it to the MITRE CAPEC framework®. This approach bridges the gap between existing cybersecurity frameworks and server log analysis, enabling a novel security pipeline. By automating threat detection and mitigation,CYBRANA enhances the security posture of digital infrastructure.**

***Index Terms*—Firewall, NIDS, Log Analysis, Random Forest, MITRE, CVSS Scoring, SIEM, SOC**

I. Introduction

In the modern-day digitally connected world, cybersecurity stays a key undertaking. The explosive growth of local vulnerabilities and complex cyber threats, makes shielding digital property a prime mission. One of the key security measures used on this community is a firewall, a key feature of community protection policy. Firewalls act as gatekeepers, screening network site visitors and aim to restrict access by maintaining confidentiality, integrity and data availability - the pillars of the CIA triad. Firewall pillars mainly depend on the ability to manifest itself and be applied to the arrival and departure of local visitors, By choosing to allow or deny visitors based on pre-set security guidelines and acting in line with the CIA Triad, a firewall works to ensure privacy with useful measures to prevent access to sensitive information, prevent unauthorized access, encrypt data fence integrity, and ensure local delivery. However, current day Firewall does not provide the required security despite it's critical necessity. Information from cybersecurity and artificial intelligence enables firewalls to evolve more aggressively in the face of growing threats, increasing the ability to operate come and respond to new attacks in real time.

The main idea behind the development of the CYBRANA framework stems from the critical shortcomings observed in conventional firewall setups. The three major challenges faced by the firewall systems include insider threats, which originates from within organisational boundaries by leveraging on a kind of psychology called social engineering and exploiting the security gaps. Distributed Denial of Service (DDoS) attacks inundate networks, disrupting regular operations. Eavesdropping exploits unsecured communication channels, while host attacks and password guessing attempts capitalise on vulnerabilities within HTTP headers and authentication mechanisms, respectively. Additionally, protocol-based vulnerabilities remain a persistent concern, undermining the robustness of communication protocols. There are multiple challenges or vulnerabilities within existing firewall systems that are intended to protect networks or systems from unauthorised access or cyber threats. Problems with firewalls can be quite disastrous to operations.

The CYBRANA framework provides a strong defence against a variety of cybersecurity attacks. By utilising the YAML files from the community-sourced projectdiscovery/nuclei repository [1] , it detects insider threats using generic attack signatures and pattern-matching. It is proficient in detecting and mitigating DDoS attacks by removing repeated request packets containing questionable content due to the fact it was trained on a Stateful Firewall. Proprietary YARA (Yet Another Recursive Acronym), and YAML rules prevent efforts at eavesdropping. Generic signatures and pattern matching are used to identify host attacks. Additionally, YARA rules are used by CYBRANA to identify protocol-based attacks in network traffic. By ensuring the protection of digital assets from ever-evolving and sophisticated cyber-attacks, this flexible and proactive strategy solidifies CYBRANA as a state-of-the-art cybersecurity solution. This framework incorporates elements from the NIST (National

Institute of Standards and Technology) guidelines and integrates Machine Learning models like random forest, trained on historical logs, to detect anomalous behaviour, providing a proactive defense mechanism against evolving cyber threats.

This paper is not simply an accumulation of various existing frameworks, rather, proposes a novel pipeline-based system where we generate security detections of indicators of compromise from server syslogs using (1) a machine-learning model trained on firewall actions (2) a vast community-contributed ruleset and correlate these detections with existing frameworks to generate a comprehensive and investigable report from the outcomes of this pipeline.

II. Literature Review

Solanki's work [2] on limiting attack surface for infrastructure applications using custom YAML templates in Nuclei Automation aligns with reducing attack surface in various systems.

[3] describes a novel architecture for automatically creating attack graphs from system topology, CAPEC, CWE, and CVE datasets. The system captures several sorts of known or possible attack patterns using adaptive metadata and generates a set of visualisations based on the assault stages. The system also connects enterprise mitigations from the MITRE ATT&CK framework to the security holes detected and interfaces with a third-party optimisation tool to help enterprises minimise security expenses.

The purpose of [4] is to discuss how to use custom YAML templates to automate vulnerability assessment and management for infrastructure applications. The author created a YAML template that incorporates the most recent infrastructure application vulnerabilities and integrated them into Nuclei [1], an open-source community-powered vulnerability scanner and used it to scan and assess the security threats of a live infrastructure application.

[5] discusses how data mining and machine learning can be used to detect network traffic irregularities in firewall records. The authors aggregate comparable firewall log entries and identify patterns of normal and abnormal behaviour using clustering and association rule mining approaches. They also classify and rate the anomalies based on their severity and frequency using classification and outlier detection algorithms.

[6] discusses how to use the Common Vulnerability Scoring System (CVSS) to increase the accuracy of vulnerability evaluation. The authors ran an experiment with 67 software engineering students to see how different information cues affected evaluation accuracy. They discovered that traditional vulnerability sources (such as CVE or NVD) provide insufficient baseline information for reliable assessment. More information on the assets, threats, and vulnerability types improves accuracy. The presence of known dangers (such as vulnerabilities or malware) reduces the accuracy.

[7] is about using the Random Forest machine learning algorithm to classify Android applications as malicious or benign based on their behaviour. The paper discusses the advantages of using the Random Forest method.

[8] discusses how to classify firewall logs and detect network assaults using supervised machine learning methods.The authors used a dataset of 500,000 Snort and TWIDS instances with four classifiers: Naive Bayes, kNN, One R, and J48. They came to the conclusion that machine learning can aid in network security and firewall settings.

[9] presents an improved Random Forest algorithm, that discusses the growth based decision trees by using intact training samples, after which it calculates the reliability scores for each test sample based on decision paths and node importance scores.

[10] identified the most effective mapping between STRIDE and CWE, while also highlighting the challenges of this process.

[11] developed a search engine to link CVEs with CAPEC^{TM}, using CWE as a bridge, and classified CVEs based on their CVSS scores.

[12] proposed a Transformer-based learning framework for automating the mapping of CVEs to CWEs, achieving high prediction accuracy.

[13] introduced a neural network model for automatically mapping CVEs to CWEs and CAPEC^{TM}, with the goal of identifying potential attack impacts and corresponding mitigation actions. These studies collectively demonstrate the potential for automated mapping to enhance cybersecurity processes.

[14] presents YARA-Signator, an automated method for generating code-based YARA rules, achieving a high F1 score and minimal false positives.

[15] introduces Yarra, an extension to C that protects against non-control data attacks, enhancing data integrity.

[16] discusses YAYA, a Description Logic system for learning complex interrelations among objects.

[17] analyses the Japanese modal adverb yahari/yappari, focusing on its discourse and interactional functions.

[18] and [19] both address the challenge of efficiently matching regular expressions, with Pasetto focusing on data filtering and Yang on high-performance architecture. [20] enhances the matching process by introducing filtering techniques, which can significantly improve efficiency. These studies collectively demonstrate the potential of regular expressions in various applications, and the importance of optimising their matching process.

III. Materials and Methods

MITRE CAPEC^{TM} is an enterprise funded with the aid of NIST to provide an information base in cybersecurity. Common Attack Pattern Enumeration and Classifications (CAPEC^{TM}) framework enables the user to recognize how attackers make the most the vulnerabilities in packages and cyber – enabled features. By XML data scraping from the reputable internet site, CAPEC^{TM} insights are stored in a JSON format file. The use of CAPEC^{TM} is to identify appropriate mitigation techniques through attack evaluation and calculating the severity of the attack with the usage of the CVSS scoring method.

YARA is another vital tool which is used for its ease in writing cybersecurity rules and its vast compatibility. YARA rules, represented in the YAML format, provide flexible and impressive sample matching aid.

An open–source repository, projectdiscovery/nuclei [1] , designed for instant vulnerability scanning, plays a vital function in the study environment. By using the customizable templates, it automates the detection of threats and vulnerabilities in web applications. These templates are saved in YAML format that offer facilitated structure and repeatable protection scans, which leads to more easier identification and remediation of risks in numerous workflows.

In the wider context of studies, the combination of CAPEC^{TM}, CWE (Common Weakness Enumeration), CVE (Common Vulnerabilities and Exposures), and CVSS gives a complete framework. CWE identifies software weaknesses, CVE serves as a platform for sharing cybersecurity vulnerability facts, and CVSS, framed through NIST's NVD (National Vulnerability Database), gives a qualitative diploma of severity. The CVSS metrics - Base, Temporal, and Environmental - help assess the severity of diagnosed vulnerabilities.

This paper proposes a system architecture where we use multiple existing frameworks, as well as implement our own Machine Learning algorithm, and create a pipeline of events correlating our ML-based as well as rule-based detections with existing frameworks to create a comprehensive, investigable report of the output.

The system works based on the server logs and the firewall logs, which provide the basis for analysis filtering. It is a multi-layer architecture involving artificial intelligence, machine learning, rule-matching, and utilisation of multiple frameworks primarily focusing on MITRE's CAPEC^{TM} and CWEs, and NIST NVD's Common Vulnerability Scoring System.

The server and firewall logs are first pre-processed and fed into the machine learning model. The machine learning model comprises 2 phases - model creation and model execution. The model is first created and stored in a .joblib file, and then is run. The model is trained using the firewall logs by feeding it with historic actions of the firewall for historic server logs. After this, the model, in its execution phase, predicts the firewall outputs of the server logs, by assigning every record an output value of "Allow", "Deny", "Drop", or "Reset". Allow denotes permitted network traffic, Deny indicates traffic denied due to suspicious behavior, and Drop signifies traffic terminated after prolonged activity. This is then passed on-to the YARA rule-matching engine.

The YARA rule-matching engine allows users to write their own custom rules in YARA or YAML. It also uses a large community-sourced database of YAML rules from projectdiscovery/nuclei [1], with over 6,000 YAML rules, and converts all rules to YAML format for ease of work. This YAML rule-matching engine then matches every record of log that is marked suspicious to a YAML rule if possible, predicting the exact methodology of the attack. Another engine simultaneously reads all the YAML files available, and matches each YAML rule to a CAPEC^{TM} (Common Attack Pattern Enumeration and Classification) tactic. This is performed as a part of an elaborate procedure to calculate the risk score of a given malicious log record. Each CAPEC^{TM} Tactic has relevant CWEs, and each CWE has relevant CVEs. These are found from the MITRE CAPEC^{TM} and MITRE CWE database, and are stored/noted in an intermediate CSV file. Each CVE, published by NIST, has a calculated CVSS score and a CVSS vector which explains the calculation of the score.

The mean of all CVSS scores of CVEs for a CWE is noted, and the mean of all CVSS scores of a CWE for a CAPEC^{TM} Tactic is noted, and this way, a risk assessment and scoring system has been established to classify and prioritise the risk levels posed by log entries matching any stored YAML rule.

The output of the rule-matching engine is then taken, and the calculated risk level scores are assigned to all malicious logs, and these logs are ordered in descending order of risk, which allows an information security officer to understand the log entries of the most risk, and identify any vulnerabilities on their system which may have been identified by users, and the IP addresses of these users if required. The system also takes into account the CVSS Vectors that are averaged and calculated for each log entry, to classify them into one of the 3 suspicion categories - Interception (Breach of Confidentiality), Interruption (Breach of Integrity), or Modification (Breach of Availability) and are presented to the user in an interactive dashboard, with a feature to trace the log records in-detail, utilising openly available APIs and tools for general reconnaissance techniques of an attacker such as IP-Geo-location lookup, Identify what application used the port attacked, etc.

This offers a next-generation solution to software-based HTTP Server Log filtering (software-based firewall) and log analysis for Security Operation Centres and Security Analysts.

IV. Results and Discussion

Recall measures a system's ability to correctly identify the action, while precision measures its accuracy in identifying the right action. A high precision score indicates good performance, while a low score indicates mis-identification. Accuracy measures overall performance, with a high accuracy score indicating good identification of both correct and incorrect individuals. The F-measure, which is the product of recall and precision, measures overall system performance, with a close value of 100% indicating better performance.

In Figure 3 we show the distribution of time taken to consolidate all rules onto YAML format. It highlights outliers, the central tendency (average time), and the spread of conversion times. This aspect is crucial for assessing the efficiency of the conversion process. Outliers might indicate complexities or issues in specific rule conversions, potentially requiring optimization or investigation within the system's rule-matching workflow.

Figure 4 visualizes the distribution of rule counts across different categories or types within the community-sourced YAML rules database. It shows the proportions of various rule categories (such as attack types, vulnerabilities, or exploit methodologies) within the total rule count. The system utilizes a large community-sourced database of YAML rules obtained from projectdiscovery/nuclei, containing over 6,000 YAML rules, which are then converted for ease of use within the system.

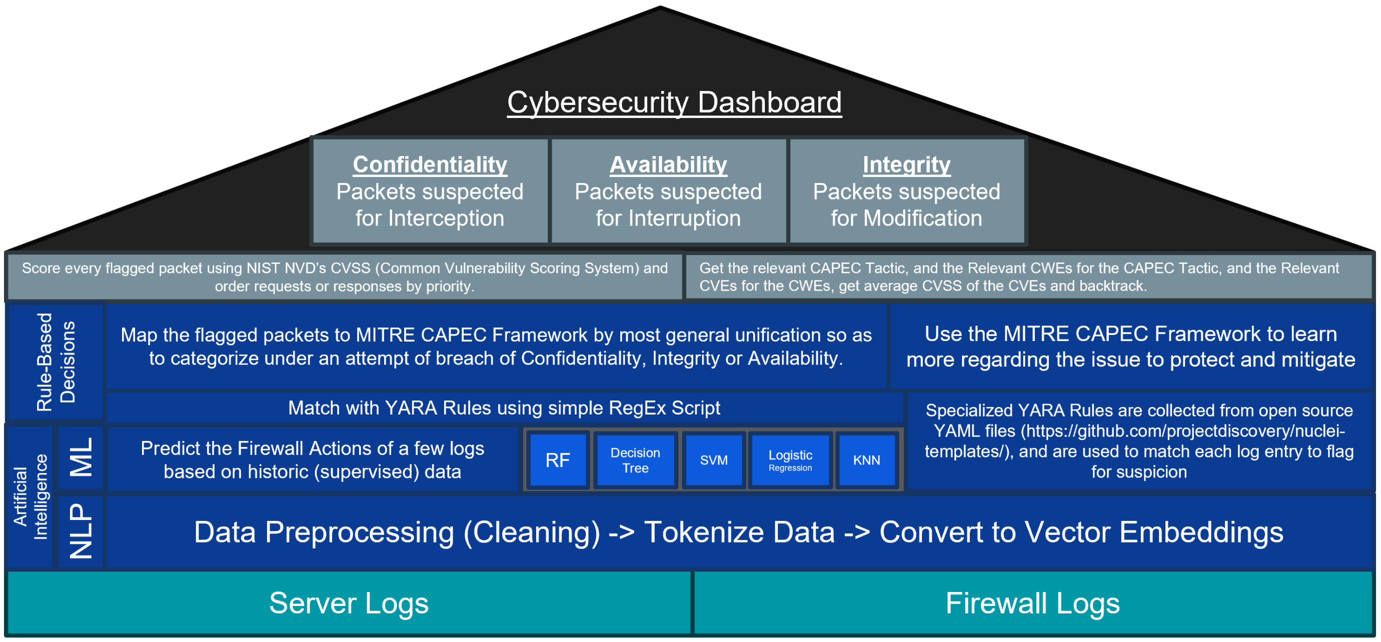
Cybersecurity Dashboard
Confidentiality
Packets suspected for Interception
Availability
Packets suspected for Interruption
Integrity
Packets suspected for Modification
Score every flagged packet using NIST NVD's CVSS (Common Vulnerability Scoring System) and order requests or responses by priority.
Get the relevant CAPEC Tactic, and the Relevant CWEs for the CAPEC Tactic, and the Relevant CVEs for the CWEs, get average CVSS of the CVEs and backtrack.
Rule-Based Decisions
Map the flagged packets to MITRE CAPEC Framework by most general unification so as to categorize under an attempt of breach of Confidentiality, Integrity or Availability.
Use the MITRE CAPEC Framework to learn more regarding the issue to protect and mitigate
Match with YARA Rules using simple RegEx Script
Specialized YARA Rules are collected from open source YAML files (https://github.com/projectdiscovery/nuclei-templates/), and are used to match each log entry to flag for suspicion
Artificial Intelligence
ML
Predict the Firewall Actions of a few logs based on historic (supervised) data
RF
Decision Tree
SVM
Logistic Regression
KNN
NLP
Data Preprocessing (Cleaning) -> Tokenize Data -> Convert to Vector Embeddings
Server Logs
Firewall Logs

Fig. 1. The conceptual architecture depicts the process of dashboard creation to let security professionals view network security posture in real time, in a bottom-up manner. Analyzing flagged packets through CVSS scoring and CAPEC^{TM} mapping enables rapid threat identification and informed risk mitigation strategies.

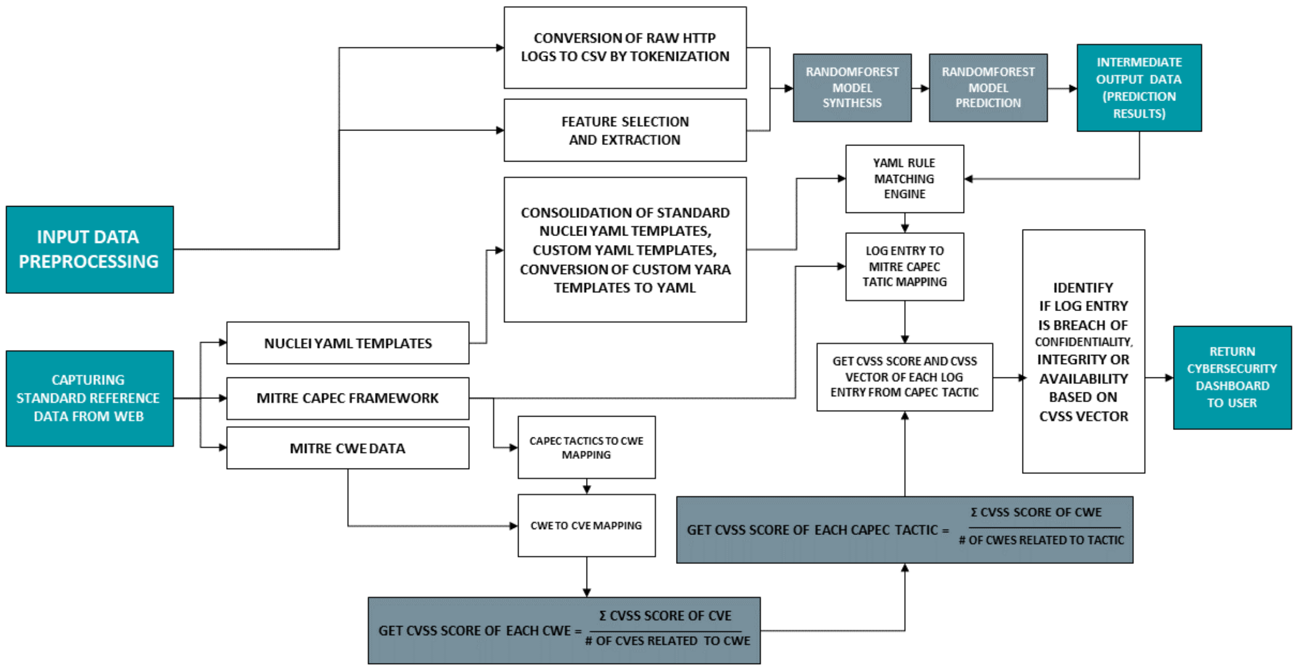
CONVERSION OF RAW HTTP LOGS TO CSV BY TOKENIZATION
FEATURE SELECTION AND EXTRACTION
RANDOMFOREST MODEL SYNTHESIS
RANDOMFOREST MODEL PREDICTION
INTERMEDIATE OUTPUT DATA (PREDICTION RESULTS)
INPUT DATA PREPROCESSING
CONSOLIDATION OF STANDARD NUCLEI YAML TEMPLATES, CUSTOM YAML TEMPLATES, CONVERSION OF CUSTOM YARA TEMPLATES TO YAML
YAML RULE MATCHING ENGINE
LOG ENTRY TO MITRE CAPEC TATIC MAPPING
IDENTIFY IF LOG ENTRY IS BREACH OF CONFIDENTIALITY, INTEGRITY OR AVAILABILITY BASED ON CVSS VECTOR
RETURN CYBERSECURITY DASHBOARD TO USER
CAPTURING STANDARD REFERENCE DATA FROM WEB
NUCLEI YAML TEMPLATES
MITRE CAPEC FRAMEWORK
MITRE CWE DATA
GET CVSS SCORE AND CVSS VECTOR OF EACH LOG ENTRY FROM CAPEC TACTIC
CAPEC TACTICS TO CWE MAPPING
CWE TO CVE MAPPING
GET CVSS SCORE OF EACH CAPEC TACTIC = Σ CVSS SCORE OF CWE / # OF CWES RELATED TO TACTIC
GET CVSS SCORE OF EACH CWE = Σ CVSS SCORE OF CVE / # OF CVES RELATED TO CWE

Fig. 2. The flowchart illustrates how our model identifies potential dangers. The HTTP Server logs serve as the dynamic key input to our automated assessment engine, whose rules, algorithms and artificial intelligence model analyze log anomalies to protect network systems against imminent threats, as well as categorize and prioritize them to display them in an interactive dashboard.

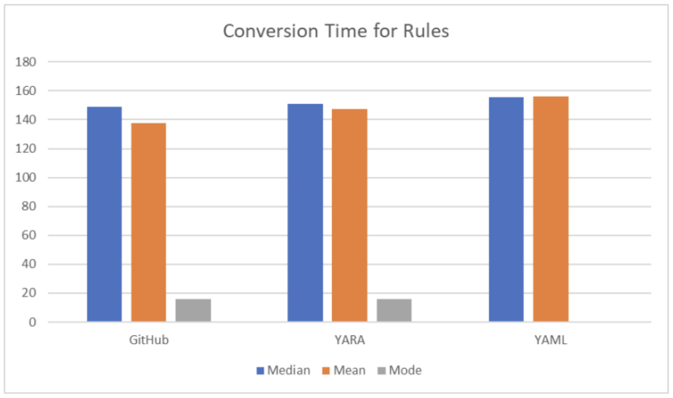

Fig. 3. Conversion Time for Rules

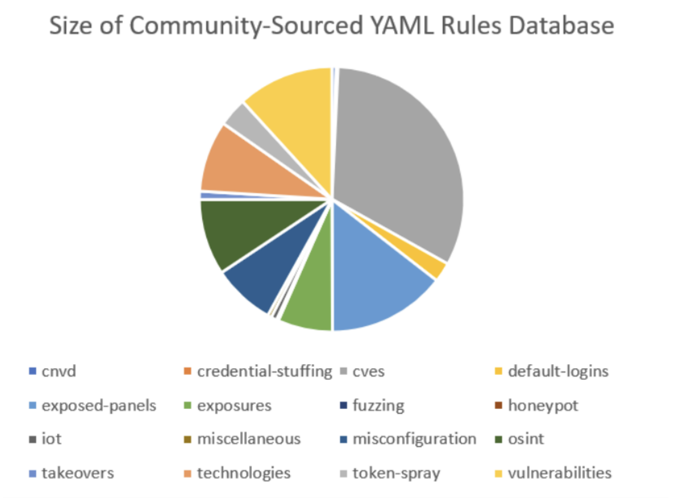

Fig. 4. Size of Community-Sourced YAML Rules Database (Pie Chart)

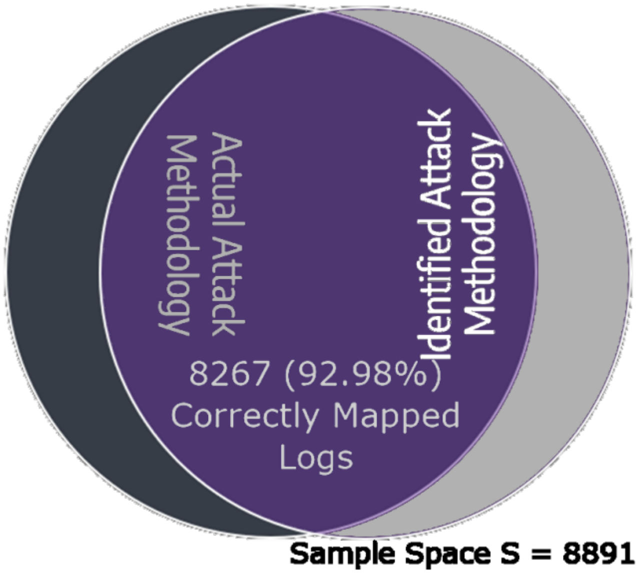
Actual Attack Methodology
Identified Attack Methodology
8267 (92.98%) Correctly Mapped Logs
Sample Space S = 8891

Fig. 5. Identified Attack Methodologies (Venn Diagram)

Figure 5 represents the overlapping representation of predicted and actual attack methodologies. It showcases the system's accuracy in predicting attack methodologies based on matched logs. The diagram illustrates the overlap between predicted methodologies and actual identified attack patterns, indicating the system's effectiveness in understanding and categorizing attack methods. In our sample set of 21685 lines of logs, we have 8891 logs that are actually malicious. Out of these 8891 logs, we have captured the attack methodologies of all logs, however, only 8267 logs map to the actual attack methodologies. This proves an accuracy of 92.98% in mapping firewall logs to MITRE CAPEC^{TM} Attack tactics, and hence proves the accuracies of their calculated risk scores.

Decision Trees are appreciated for their simplicity and ease of interpretation, making them useful for initial insights into patterns in cybersecurity logs. However, they are prone to overfitting, which is very common in case of cybersecurity logs due to it's sheer size. They also have tendency to memorize noise in the data, reducing their ability to generalize to new, unseen attack patterns effectively.

SVM (Support Vector Machine) excels in handling complex data by mapping it into higher-dimensional spaces, which can enhance the separation between malicious and normal network traffic. However, SVM can be computationally intensive and less effective with large-scale datasets commonly found in cybersecurity, due to memory constraints and the time required for training on extensive feature sets.

Logistic Regression provides probabilities for binary outcomes, whether the given log is safe or not. Nevertheless, its assumption of linear relationships between input parameters and the log-odds of the response variable may limit its effectiveness in capturing nonlinear relationships and intricate attack patterns present in cybersecurity logs like in our case where we go a step furhter by sample the logs as allow, deny and drop.

K-Nearest Neighbors identifies patterns based on the proximity of data points in feature space, making it suitable for detecting anomalies, like deny and drop, in non-linear data distributions typical in cybersecurity. However, its performance can degrade in high-dimensional datasets due to computational costs and sensitivity to irrelevant or redundant features, which are common in diverse cybersecurity logs.

In our Random Forest (RF) model, we are using parameters like ip_address, datetime, http_method, url_resource_path, start_date, end_date, http_version, http_status_code, bytes_sent, user_agent, and IDS_decision. The IDS_decision acts as a tags, categorizing which of the log entries in the dataset are safe and unsafe for supervised learning [23]. The RF model is able to investigate complex correlations and interactions between various variables which is quite difficult and resource intensive in case of other models on trials as each combination had to be trained and created before we can test the model's accuracy with the current relation between

TABLE I
PERFORMANCE COMPARISON FOR DIFFERENT MODELS

	Decision Tree	SVM	Logistic Regression	K-Nearest Neighbors	Random Forest
Recall	0.891	0.882	0.880	0.886	0.898
Accuracy	0.891	0.882	0.880	0.886	0.898
Precision	0.881	0.882	0.778	0.841	0.891
F-Measure	0.886	0.882	0.826	0.863	0.894

TABLE II
CONVERSION TIME FOR RULES W.R.T FIGURE 3

	Central Tendency			Standard Deviation	
	Median	Mean	Mode	Population Standard Deviation	Sample Standard Deviation
GitHub	149	137.709	15.625	46.638	46.641
YARA	151	147.241	15.625	33.399	33.425
YAML	155.5	156	N/A	11.683	13.490

TABLE III
CONFUSION MATRIX FOR RANDOM FOREST MODEL

	Predicted: Allow	Predicted: Deny	Predicted: Drop
Actual: Allow	495	7	0
Actual: Deny	15	33	0
Actual: Drop	1	0	18

parameters . But in the case of RF model, it efficiently recognizes complex attack patterns and differentiates them from regular network traffic by taking into account all potential combinations automatically and discarding the irrelevant ones in the process. As seen in Table I, the RF performs better than other models such as Decision Trees, SVM, Logistic Regression, and K-Nearest Neighbors due to its capacity to handle a variety of data sources and identify complex correlations among parameters. In the end, the RF's capacity to fully integrate and understand these characteristics improves the accuracy of its cybersecurity threat detection and classification.

Now, transitioning to Table III which gives us the confusion matrix for the RF model on a small sample classifying the logs into three categories, namely Allow, Deny and Drop. The confusion matrix played a very important role in determining the effectiveness of different models. In the context of CYBRANA, True Positive(TP) represents the instances where the RF model accurately predicts all the 3 classes and the actual data corroborates with this prediction. For example, there are 495 instances where ”Allow” was predicted correctly. Similarly, for True Negatives(TN), the model correctly predicts 33 and 18 instances of ”Deny” and ”Drop”. Whereas, False positive (FP) and False Negative (FN) give us the instances where the model's prediction is not matching with the actual label.

V. CONCLUSION

This paper describes an attack detection methodology using machine learning algorithms and natural language processing (NLP). It uses web scraping to extract data-breach-based resources and appropriate mitigation measures. The AI model has been trained on random forest classification against a server log. Threats detected by the AI have been examined against a set of YAML rules to determine the type of attack. The identified attack has been mapped to the MITRE CAPEC^{TM} rule set using Regex-based rule matching using python scripting to determine attack severity, prevention, and mitigation techniques. The user receives a comprehensive report on the attacks and how to mitigate future attacks. It helps organisations reduce the impact of a data breach and take the necessary steps to protect themselves, their employees, clients and others from potential attacks.

The proposed framework derives it's root inspiration from works [2], [3] and [4] utilizing predefined custom YAML rules and CAPEC Framework to aid in threat detection and mitigation. The existing system is taken a step further by introducing an AI model based on Random Forest Classification that automates the work of matching YAML rules with the identified request paths. This not only reduces the time indulged in searching through an extensive set of rules, but also makes room for intelligence. The proposed framework generates a set of hypothesis based on the training data which helps in detecting both known and unknown threats based on the model's inferences.

In future scope, the system can be linked to a file where the real-time logs are output for both firewalls and server syslogs. The firewall logs can be accumulated in batches of relevant size to updating the deployed machine learning model with new data, as a part of Continuous Training (CT) in MLOps [21] for Enhancing the Accuracy of Machine Learning Models using DevOps, Continuous Integration, and Continuous Deployment [22]. A file integrity monitoring tool (watchdog) can be implemented by linking the server syslog files to the application, and on detecting change events, the latest line can be run through CYBRANA to ensure real-time security systems.

Simultaneously, a generative artificial intelligence model could also be trained in a reinforcement cycle based on the server syslog, firewall's response, and the existing YAML rulseset, to automatically generate new and relevant YAML rules in order to keep a continuous enhancement of cybersecurity in the organization.

References

[1] 2023. [Online]. Available: https://github.com/projectdiscovery/nuclei

[2] H. Solanki, Limiting Attack Surface for Infrastructure Applications using Custom YAML Templates in Nuclei Automation, National College of Ireland, Ireland, 2023.

[3] C. F. Ö. Sönmez, P. Hankin, . Malacaria, C. Capec, C. Databases, and . ', "Attack Dynamics: An Automatic Attack Graph Generation Framework Based on System Topology," Computers & Security, vol. 123, 2022.

[4] H. Solanki and Vinod, Limiting Attack Surface for Infrastructure Applications using Custom YAML Templates in Nuclei Automation, Dublin, 2023.

[5] R. Winding, T. Wright, and M. Chapple, "System Anomaly Detection: Mining Firewall Logs," Securecomm and Workshops, pp. 1–5, 2006.

[6] L. Allodi, S. Banescu, H. Femmer, and K. Beckers, "Identifying Relevant Information Cues for Vulnerability Assessment Using CVSS," in 2018 ACM Conference on Data and Application Security and Privacy (CODASPY), 2018, pp. 119–126.

[7] M. S. Alam and S. T. Vuong, "Random Forest Classification for Detecting Android Malware," in 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber. Physical and Social Computing, 2013, pp. 663–669.

[8] S. D. H. E. As-Suhbani and Khamitkar, "Classification of Firewall Logs Using Supervised Machine Learning Algorithms"," INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING, vol. 7, 2019.

[9] Y. Zhang, L. Luo, X. Ji, and Y. Dai, "Improved Random Forest Algorithm Based on Decision Paths for Fault Diagnosis of Chemical Process with Incomplete Data," Italian National Conference on Sensors, vol. 21, pp. 6715–6715, 2021.

[10] A. Honkaranta, T. Leppänen, and A. Costin, "Towards Practical Cybersecurity Mapping of STRIDE and CWE - a Multi-perspective Approach," Conference of the Open Innovations Association, vol. 29, pp. 150–159, 2021.

[11] Χρηστὸς Γρηγοριάδης, Identification and assessment of security attacks and vulnerabilities, utilizing CVE, CWE and CAPEC, Piraeus, Greece, 2019.

[12] S. S. Das, E. Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer, "V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities"," IEEE DSAA, vol. 8, no. 1, pp. 1–12, 2021.

[13] E. Aghaei, E. Al-Shaer, and . . Threatzoom, 2019.

[14] F. Bilstein, D. Plohmann, and . . Yara-Signator, Automated Generation of Code-based YARA Rules, Bordeaux, France, 2019, vol. 5.

[15] C. Schlesinger, K. Pattabiraman, N. Swamy, D. Walker, and B. G. Zorn, "Modular Protections against Non-control Data Attacks," Proceedings - 24th IEEE Computer Security Foundations Symposium, pp. 131–145, 2011.

[16] J. Alvarez, "A Description Logic System for Learning in Complex Domains," in Proceedings of the 1998 International Workshop on Description Logics (DL'98), 1998.

[17] K. Senko, "Discourse and Interactional Functions of the Japanese Modal Adverb Yahari/Yappari," Language Sciences, vol. 13, no. 1, pp. 39–57, 1991.

[18] D. Pasetto, F. Petrini, and V. Agarwal, "Tools for Very Fast Regular Expression Matching," " in Computer, vol. 43, no. 3, pp. 50–58, 2010.

[19] Y. Yang and V. Prasanna, "High-Performance and Compact Architecture for Regular Expression Matching on FPGA," IEEE Transactions on Computers, vol. 61, no. 7, pp. 1013–1025, 2012.

[20] T. Qiu, X. Yang, and B. Wang, "Filtering Techniques for Regular Expression Matching in Strings," DASFAA Workshops, pp. 118–122, 2018.

[21] "An Open-Source and Portable MLOps Pipeline for Continuous Training and Continuous Deployment," University of Helsinki Open Repository, 2023. https://helda.helsinki.fi/bitstream/10138/356945/2/Luo_Yumo_thesis_2023.pdf (accessed Jan. 18, 2024).

[22] M. Y. S. Krishna and S. K. Gawre, "MLOps for Enhancing the Accuracy of Machine Learning Models using DevOps, Continuous Integration, and Continuous Deployment," Research Reports on Computer Science, pp. 97–103, Jun. 2023.

[23] National Remote Sensing Centre (NRSC) at Hyderabad, Hack2Skill. (n.d.). Framework Model for Identifying, Detecting and Reporting for Cyber Security on Bhuvan Portal [Online]. Available: https://drive.google.com/file/d/1dFrkkpBioaS8WbPRFNxAksWS1TsHiPpO/view