SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

YARWEB: "*WEB-BASED GENERIC YARA RULE GENERATOR*"

Mr. Shreyas Biju Nair[1], Mr. Laalas Tadavarthy[2], Mr. Kailas M K[3], Mr. Gowrishankar T O[4]

[1]*Computer Science and Engineering, Presidency University, Bangalore, India*
[2]*Computer Science and Engineering, Presidency University, Bangalore, India*
[3]*Computer Science and Engineering, Presidency University, Bangalore, India*
[4]*Computer Science and Engineering, Presidency University, Bangalore, India*

Article DOI: https://doi.org/10.36713/epra15953
DOI No: 10.36713/epra15953

ABSTRACT

In the modern 21st century, surfing the internet has become difficult due to the rise of malware and adware. Sensitive information is often a risk to be stored without encryption. If malware does infiltrate, devising a solution to mitigate the risks is difficult and tiresome. The proposed framework presents a web-based approach to automatically generate a YARA rule for a malicious file uploaded by the user. Since it is a search engine-based model, it becomes extremely portable and useful. The testing of this prototype depicts that it is fully capable of detecting malicious samples with an average accuracy of 0.80.

KEYWORDS—*Malware Analysis, YARA Rules, Generic Rules, Malicious Strings, Fully Automated.*

I. INTRODUCTION

The cyber realm is rife with malicious programs, often referred to as malware, designed to exploit vulnerabilities and inflict harm upon unsuspecting systems. Recent years have witnessed a surge in the quantity, diversity, and sophistication of these threats, necessitating automated solutions for their analysis and mitigation [1].

Among the various methodologies proposed, signature-based detection, despite its limitations, remains a cornerstone of endpoint security due to its simplicity, speed, and effectiveness. Anti-virus software leverages signature matching by maintaining a database of known malicious patterns (signatures). During scans, files are hashed (e.g., using MD5) and compared against this database. A match triggers alarm bells, flagging the file as a potential threat. However, this approach hinges on frequent database updates and comprehensive signature coverage for every new malware variant [2]. The inherent sensitivity of cryptographic hashes to minor code alterations can render them ineffective against polymorphic malware strains. Maintaining and updating vast databases of known signatures also poses logistical challenges.

Seeking to address these limitations, researchers have explored alternative signature-based techniques like string-based and rule-based approaches. YARA, a popular framework, exemplifies this effort, offering a powerful pattern-matching engine for scanning large datasets. While effective, YARA relies on meticulous rule creation, demanding expert knowledge and extensive malware analysis experience. The upkeep and adaptation of these rules further add to the complexity.

This work aims to bridge this gap by automating the generation of effective YARA signatures for known malware samples. The objective is to develop signatures capable of identifying new malware variants with high accuracy (precision and recall) while minimizing false positives [3]. The proposed methodology focuses on generating signatures for Microsoft Windows executables.

The subsequent sections delve deeper into these topics: section II provides all the details about YARA rules and its format. Section III provides research-based studies of multiple existing works. Section IV depicts a comprehensive comparison of YarWeb with its competitors. Sections V and VI revolve around the proposed framework and extended login controls respectively. Section VII reveals the intricacies and usefulness of the results generated. Section VIII concludes the research paper.

II. YARA RULES

YARA rules serve as meticulous blueprints for identifying malicious code. They operate by seeking out specific patterns—known as signatures or strings—within files, folders, or processes, and comparing them to those associated with documented malware [4]. These rules are composed of three essential components: meta, strings, and condition. Meta is the section that provides descriptive information about the rule, such as its author, purpose, and creation date. Strings are the heart of YARA's detection as the capabilities of the rule lie within its string definitions. These strings fall into three distinct categories: Text, hexadecimal, and regular

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

expression strings [5]. Text strings are readable text sequences, potentially enhanced with modifiers like nocase (case-insensitive matching), ASCII, wide (Unicode), and fullword (whole-word matches) for refined control [6]. Hexadecimal strings represent raw byte sequences, offering flexibility through wildcards (?), jumps ([n-m]), and alternatives (|) to accommodate variations [6].Regular expression strings, introduced in YARA 2.0, extend pattern-matching capabilities beyond straightforward text and hex sequences, enabling intricate matching logic [6]. Condition is the final section that acts as the gatekeeper, determining whether a rule fires or remains dormant. It specifies the minimum number of strings that must successfully match within a target file to warrant a malware classification [7]. The condition itself is expressed as a Boolean expression, mirroring those found in common programming languages [6].

By meticulously crafting these rules, security professionals can arm YARA with the knowledge necessary to pinpoint malware amidst vast datasets, empowering swift and effective threat mitigation.

```
rule YaraExample
{
   meta:
         author = "YarWeb"
         date = "28.12.2023"
         version = "1.0"
   strings:
         $a = "this is a malicious string"
         $b = {6A 40 68 00 30 00 00 6A 14 8D 91}
   condition:
          all of them
}
```

Fig. 1. YARA Rules: Example

III. RELATED WORK

Automatic generation of YARA rules became the most sought-after method because generating rules manually by analyzing strings becomes a tedious and time-consuming task. This led to the development of tools that perform the previously mentioned task with efficiency. Here, three such tools are explained: yaBin, yaraGenerator, and yarGen.

A. yaBin

Within the arsenal of YARA rule generation tools, yaBin stands out for its unique approach to identifying malware. Developed by the Alien Vault Open Threat Exchange (OTX) community, this Python-based tool zeroed in on rare functions lurking within malware samples or families [8]. yaBin meticulously scans code for function prologues, which are telltale markers that signal the beginning of a function. It's akin to spotting the opening lines of a chapter in a book. To ensure focus on the truly distinctive, yaBin cross-references identified strings against a comprehensive whitelist of frequently used library functions. The resulting YARA rules are crafted as lists of hexadecimal strings, capturing the unique byte sequences associated with rare functions. When scanning suspected malware files, yaBin compares these fingerprints, seeking similarity in their byte-level composition. A close match raises a red flag, indicating a potential malicious presence.

B. yaraGenerator

Developed by Chris Clark, yaraGenerator takes a distinct approach to crafting YARA rules, prioritizing adaptability to different file types [9]. This Python-based tool recognizes that malware can infiltrate systems through various disguises. yaraGenerator meticulously analyzes code, identifying strings that hold the potential to distinguish malicious code from benign files. It then leverages code refactoring techniques to streamline these strings, enhancing their effectiveness as malware fingerprints. To ensure focus on identifying truly unique characteristics, yaraGenerator employs a blacklist of roughly 30,000 common strings, organized by file format. This filter eliminates strings that frequently appear in legitimate software, reducing the likelihood of false positives. The resulting YARA rules are crafted with sensitivity to the specific file format being examined. This attention to detail ensures that the signatures effectively capture the unique patterns that often signal malware within each distinct file type.

C. yarGen

yarGen stands apart for its innovative use of machine learning and natural language processing techniques. It employs fuzzy regular expressions, a Naive Bayes classifier, and a Gibberish Detector to meticulously analyze code, discern patterns, and pinpoint those strings and opcodes most likely to signal the presence of malware [10]. To ensure precision, yarGen cross-references identified strings and opcodes against extensive databases of known, legitimate software. This filtering process eliminates common elements, leaving behind the more distinctive patterns that often characterize malicious code. The resulting YARA rules feature a carefully selected set of strings and opcodes, typically capped at 20 to maintain operational efficiency. These elements are chosen based on their assigned scores, reflecting their potential for accurately identifying malware.

IV. OBJECTIVES

YarWeb is a unique tool that automatically generates generic YARA rules of a specific malware signature on a web-based platform. The objectives that this project aims to achieve or have already achieved are listed below:

- Web-based: YarWeb's front end is entirely built on HTML, CSS, and Javascript. Being web-based is easier to navigate and use when compared to Command-line based tools.
- Login controls: This is the only YARA-related tool in the market that has production-ready features like a login page. YarWeb ensures that users are allowed to create their profiles and their work is isolated from one another.

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

- Admin functionalities: Adding to the production-ready environment, an admin has the capabilities to monitor other users and provide input.
- Rule Updation: Rules created or existing in the database can be uploaded on YarWeb without a hassle.
- Portability: Since the project is web-hosted, it is OS-independent and only minimal Python packages are needed for YarWeb to function.
- Fast processing: YarWeb works on string matching and concatenation. This ensures that the YARA rule is produced within seconds.

IV. COMPARISON WITH YARWEB

Even though these tools automatically produce YARA rules, there are major drawbacks to them. Some of these drawbacks are:

- These tools are dependent on a command terminal to work. They are to be installed and commands are to be remembered for the functioning of these tools.
- They are not portable. Installing dependencies and packages is time-consuming.
- Processing of these rules requires large databases to be downloaded in the host server. This takes up space.
- Most of these tools take up to 10 seconds or more to produce their rules, which makes them less efficient.
- Some of these tools do not create a generic rule for a particular signature. They tend to be very specific to a file and its extension.
- They are not production-ready. They are programs working on Python scripts and do not hold any other corporate or production-based association.

YarWeb meets all these above-mentioned conditions and does more. Here, yarWeb is comprehensively compared with a relevant competitor-yarGen.

TABLE I: COMPARISON BETWEEN YARWEB AND YARGEN

Features	YarWeb	yarGen
Web Based	Yes	No
Precision	0.80	0.793
Portable	Yes	No
Processing Time	3 seconds	34 seconds
Ready to be integrated to a larger software	Yes	No
Cross-platform	Yes	No
Login/Admin controls	Yes	No
Test the produced rule	Yes	No
Update the produced rule	Yes	No
Check if the file is malicious or not	Yes	No
User interaction	Interactions with GUI/Web platform	Commands are used
Produces extra rules (super rules) which causes redundancy	No	Yes

V. PROPOSED FRAMEWORK

YarWeb is built entirely on Python and is completely open source. This makes the code easy to review and extend. Since it works primarily on the web, web-based technologies are used, such as HTML, CSS, Javascript, and JQuery. The integration between the scripts and the HTML templates is done using Flask.

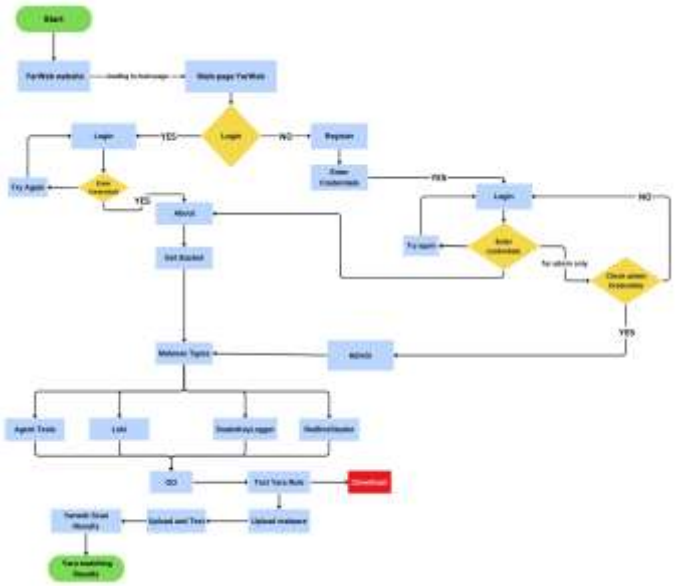

Fig. 2. Working of YarWeb

As shown in Fig. 2, a linear approach is followed to get to the main functioning of the product. This process can be broken down into 7 subsequent processes: Creating databases, Information retrieval, Login process, Admin controls, About the product, Creating the rule, and Testing the rule. Here, these functions are explained individually.

D. Creating Databases:

This process involves two steps: Collecting malware samples and Extracting malicious strings from the samples, as shown in Fig. 3. YarWeb sources its samples entirely from MalwareBazaar. This platform has a database containing various types of samples wherein a particular sample can be searched by a hash (MD5, SHA256, SHA1), imphash, tlsh hash, ClamAV signature, tag, or malware family.

This first stage involves the selection of the most commonly spotted malware signatures. Once a signature has been decided on, malware samples are downloaded on the website in a sandbox environment. It is important to download the files on a virtual/sandbox environment so that the host system is isolated from the off chance of an accidental activation of the live malware. Similarly, samples for other such signatures are collected.

The second stage involves extracting the malicious strings using tools like PEStudio. PEStudio helps to identify unknown/unique strings in the file system that may/may not

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

be malicious. These could be strings or hex codes. Once these strings are noted, they are recorded in a .csv file. YarWeb's databases are these .csv files, mostly containing only two columns: Malware_sha256 (name of the malware) and Malicious Strings. The larger the database, the more accurate would be the rules produced. .csv files are preferred because they are easier to read since the data is presented in a tabular manner, and also because pandas often support only files of that extension.

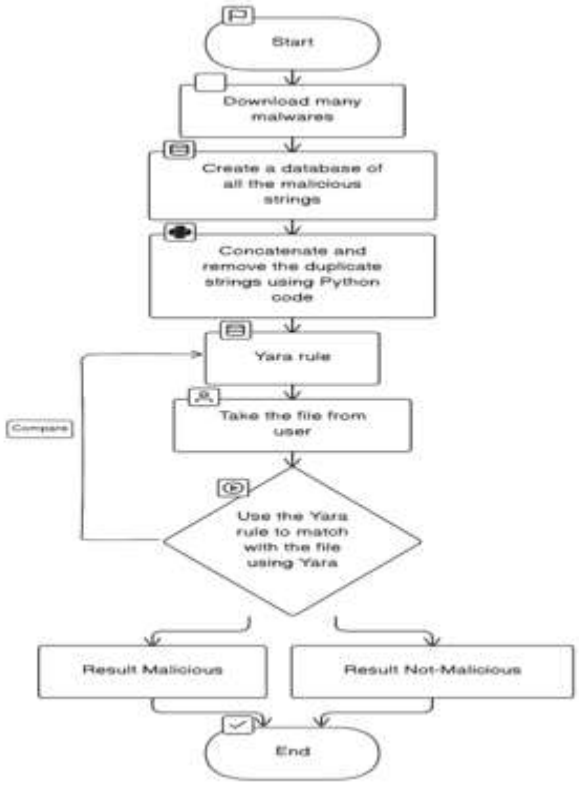

Fig. 3. Creation of database and other steps

E. Information Retrieval page

Creating the database is an entirely developer-related step. As a user/customer, YarWeb initially displays a landing page that provides all the necessary fundamental information on YARA rules and how it functions. This page is purely built for information consumption and retrieval from a third-party point of view. The webpage is fluid and interactive with rolling gifs and a professional backdrop. Throughout the project, one page leads to the other through buttons. Once the customer clicks on the Login button, it will lead to the Login page subsequently.

F. Login page

The login page contains two important tabs: For existing users and new users. Existing users can log in using their credentials whereas new users will be led to a tab wherein they are allowed to fill out the form to create credentials. These functionalities will be explained in detail below:

New Users

These users will have to fill in four important details to create their accounts: Name, Email, Password, and Confirm Password. These input fields are created using HTML and validated using Javascript. This means the email field will check whether the string inputted is in the standard email format. The password field is also validated, hence it follows the basic password requirements. If these input field requirements are not met, the user will not be allowed to proceed with the process. The Python script imports the SQLAlchemy package, which then produces a users.db file in the host's environment. All the credentials created will be stored in this .db file. YarWeb ensures that the password is not stored in plaintext. An import package, bcrypt, is an adaptive cryptographic hash function for passwords. This adds a layer of security to the entire project.

Existing Users

These users will have to fill in two important fields to log in: Registered email ID and password. Input validations are applied in this tab as well. When the user inputs their credentials and clicks on the Login button, a backend process is triggered. The credentials are checked in a linear one-to-one manner. First, the program identifies the user's email and searches the users.db database file to flag it. If the email is not found, the flag is returned as null, and the user is displayed an error message. However, if the email matches with the list of email IDs in the database, then the flag is returned as True or 1. After this, the program then checks if the passwords are matched. The password that was inputted by the user is then hashed using bcrypt. This hash value is compared with the existing hash in the database. If the values match, another flag is returned and the user will be logged in. The user is then led to a page that displays all the information about YarWeb, as shown in Fig. 4. Python script also includes session management using cookies to ensure that after the user logs in, it is impossible to return to the login page until and unless the session key expires. This prevents unnecessary form submissions and multiple logins.

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

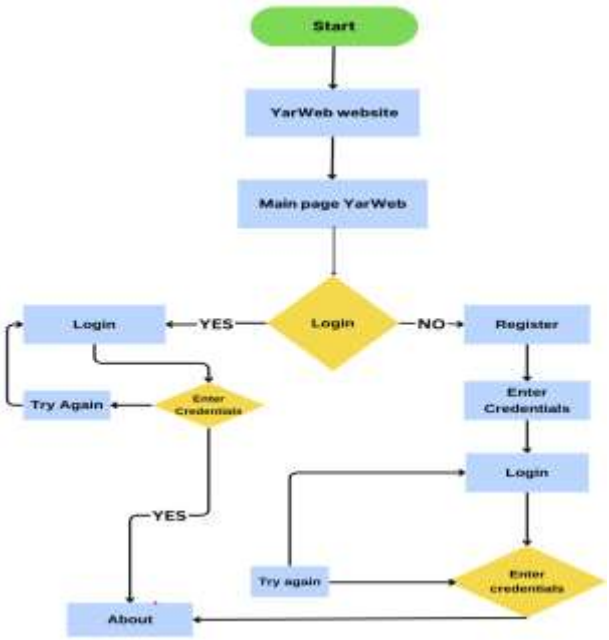

Fig. 4. User/Customer's login and workflow

Admin User:
The admin will have additional controls that are inaccessible to the regular users. Moreover, the login system does not check the database for admin credentials as they are hardcoded in the Python script. This isolates the admin credentials from losing control over the program when an unfortunate data breach occurs because the credentials will not be a part of the user database. Admin also gains specific controls: User interaction monitoring and Rule updation, as shown in Fig. 5. Both these features are located on a single platform. The Dashboard has a minimal, well-organized layout that ensures all the features are easily accessible and monitorable.

The first part of the dashboard is the User interaction monitoring system. Here, the details of the users are specified. The fields mentioned are Username(email), Number of Logins, and the total time spent on the framework. Information about the users is easily identifiable as they are displayed in floating containers, coded entirely using CSS. The number of logins field shows the cumulative figure of the number of successful logins, not the number of login attempts. This works by using an incremental count variable which increments every time the login program returns a flag positive during the login process. However, the admin is not given the control to delete the user or read their credentials, as this would be unprofessional and against an Information Technology Administrator's etiquette.

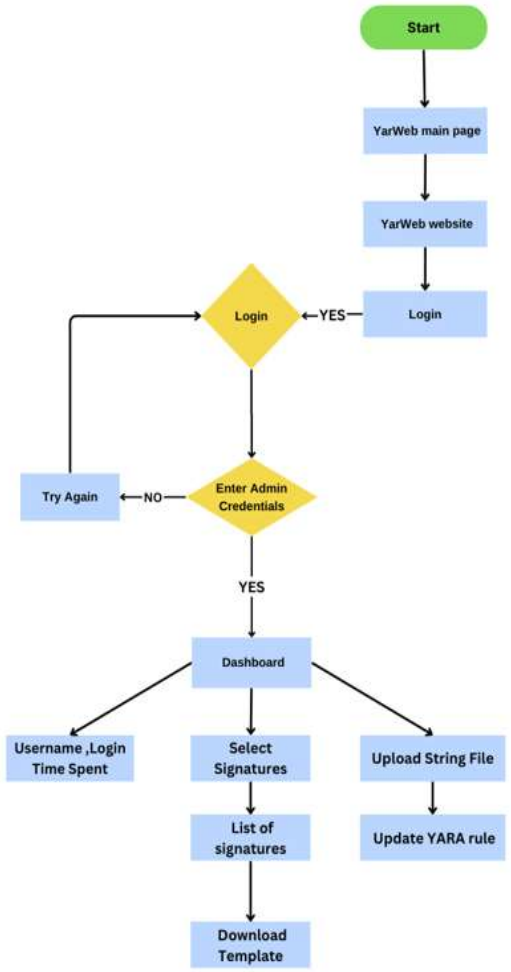

Fig. 5. Admin's login and workflow

The second part of the dashboard is all about Rule Updation. This provides the functionality for the admin to update the existing YARA rules on the go. YarWeb does this, by letting the admin pick a signature from the list of recorded malicious signatures. Once the signature has been decided on, the admin can download a template. This template is a text file that depicts the format for the strings required for the updation. The admin can then edit the downloaded template, and modify it with strings. Then this file can be uploaded to the same section using a direct Upload button. In the backend, the Python script appends these strings at the end of the existing strings in the rule. This is done so by looking for the string "condition:" since this happens to be the attribute present after the strings. After locating this, the strings from the template are concatenated and saved. The aim of this feature is quick-updation. If the YARA rule of that signature were very lengthy, it would be difficult and time-consuming for the admin to search through the existing strings to add strings of choice. Auto-save ensures that the rule is updated and saved without manually clicking on the Save button of the file. Since the file name of the YARA rule does not change from this process, the updated YARA rule can be

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

directly used for rule testing against a particular malicious file. Even though the template provides a format of a singular string, it is possible to modify it to numerous strings which can be added to the rule. The string given in the template is just for format reference, there is no cap on the number of strings.

G. About the product page:

This page serves the purpose of educating the user about YarWeb and its functioning before using the application. Information is structured in an informative and minimal manner. Complicated and technical information is not specified here. The webpage's User Interface (UI) is similar to the UI used in the landing page (A). The only difference is that this page can only be accessed after a successful login from either a user/customer or an admin. The "Get Started" button leads to the next page.

H. Creating the rule:

This is the heart of the proposed framework. The key aspect of this stage is the production of the YARA rule. Initially, the user/admin will be greeted with a plethora of malicious signatures to choose from. These signatures are displayed in containers of their own, with distinct images for easy identification. This is done entirely on CSS. Currently, four extremely common malicious signatures are used for rule generation: AgentTesla, SnakeKeyLogger, RedLineStealer, and Loki. Agent Tesla, a . NET-based keylogger, lurks in the shadows, silently recording every keystroke and capturing sensitive data as the user navigates the digital world. Its insidious capabilities extend far beyond mere keylogging, posing a significant threat to both personal and organizational security. RedLineStealer operates like a skilled pickpocket in the digital realm, covertly snatching sensitive data from a wide range of sources. It is to be noted that YarWeb is not limited to these signatures, multiple signatures can be added, provided databases are created for the same.

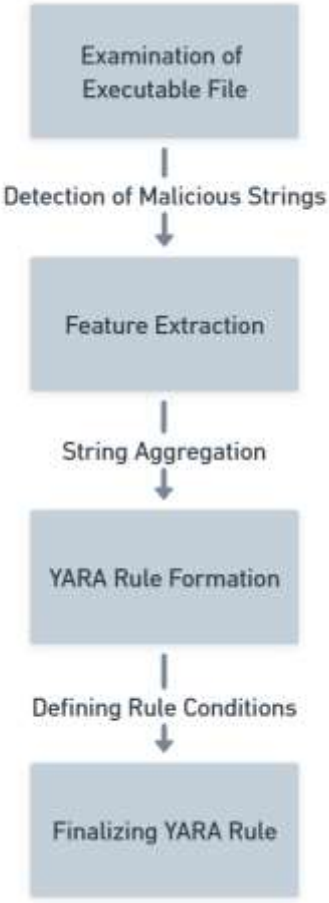

Fig. 6. Steps followed by YarWeb to create a YARA rule

Once the signature has been decided on, a special container for that particular signature appears on the same webpage. Within seconds, a YARA rule is created for that particular signature. The steps depicted in Fig. 6 are followed for the rule creation. The backend Python script locates the database of the chosen signature and reads the malicious strings column (second column). Then the program performs two special functions on the strings: Concatenation and Removing duplicates. Initially in the database, strings of each malware are stored in variables of their own respectively in the format "$sXX". This is a variable name where XX depicts a two-digit number. Each malicious file in a database contains up to 20 malicious strings that it is associated with. This means that a range of $s1 to $s20 is used to store strings for a particular malware file of a specific signature. There could be numerous such files in a database with the same variable names for their strings. To prevent the redundancy of the variable names in the final YARA rule, duplication removal is done. The script specifically reads every value following the characters "$s" to ensure that the number assigned after those characters, is unique and not previously used. Once the duplicates are removed, these strings are then concatenated to form a single set. The script also ensures that the output follows the standard YARA rule format (Fig. 1). It fills

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

default information specific to YarWeb in the Meta section of the rule. The concatenated and treated strings are added to the Strings section. A unique signature-specific condition is also listed in the rule. After the YARA rule has been created, it is stored locally in the host for further updation or usage. It is also presented in a downloadable manner on the webpage as a hyperlink. The user can click on this link to further download the YARA rule. Right under this, a "Test" button would lead to the Testing page.

I. Testing the rule:

This is the final stage of YarWeb, where the automatically generated YARA rule is put to test. On the page itself, there is a Upload button. This button is formatted to only allow executable (.exe) files to be chosen. It is to be noted, that this feature can be extended wherein any file type can be chosen. One important feature is that this page does not ask the user/admin to select a signature. This is because the script for the current and the previous page works on session keys. When the user/admin generates a YARA rule, a session is created for that specific rule. Essentially, the program remembers the most recently created rule and for which signature it was created. Due to this, the testing page automatically assumes that the testing is being done for that specific YARA rule. This feature was added to remove the redundant task of selecting a feature.

Once the file has been uploaded, pattern matching takes place. This is done using the tool yara-python. This tool is the HTML-enabled version of the YARA tool since the regular version is entirely command terminal dependent. Yara-python is programmed to analyze strings of a given file and match it against a given YARA rule. Hence, in YarWeb, yara-python considers the user/admin inputted file as the file to be analyzed, and the most recently produced YARA rule as the rule to be matched against. If the rule has matched or the strings of the rule display similar characteristics to the strings of the file, then the file is considered malicious for that particular malware signature. If the uploaded file is not a part of that specific malicious signature, then the YARA rule will not be matched and a negative result will be displayed. To ensure that YarWeb informs the user that the file uploaded is malicious, no matter the signature-based matching, VirusTotal API is integrated. VirusTotal is a popular platform used by security analysts to check whether a hash, file, or URL is malicious or not. An API key is used to fetch its capabilities on YarWeb. The file uploaded by the user on YarWeb is sent to VirusTotal using the API key. The file is scanned against multiple third-party antivirus vendors. If these vendors have reported the file as malicious, these reports will be displayed on YarWeb. The YARA rule matching of YarWeb occurs simultaneously. Both these outputs are displayed together, to prevent any waiting time. Around 60-70 reports are produced on YarWeb through the API so that the user/admin is aware of the potential dangers of the file.

VI. RESULTS AND DISCUSSION

One of the key highlights of YarWeb's results is the limited time taken to produce the outputs/YARA rules. This is achieved solely due to the simplicity of the datasets. The more complex a dataset is, the longer the result will take to form. YarWeb follows a two-column dataset, and its linear concatenation and duplicate removal technique are additional features that contribute to the quick-loading time. YarWeb loads the rules in about 3 seconds whereas the competitors load the rules in about 20-35 seconds as depicted in Fig. 7.

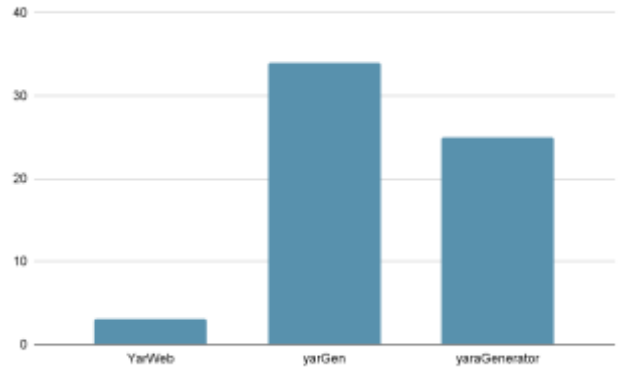

Fig. 7. Graphical representation of the time taken by YarWeb and its competitors to produce YARA rules automatically

When programming models are compared, accuracy is a great factor to measure. Accuracy is the percentage of correct classifications that a trained model achieves, i.e., the number of correct predictions divided by the total number of predictions across all classes. For YarWeb, accuracy was tested by taking 20 malware samples of the same signature, in this case, AgentTesla. A YARA rule, already generated by YarWeb, is then used against these malware test samples one by one. The results were recorded. The outcome of this test was that 16 of these malware samples were matched positive as malicious and of the same signature, 4 of them were considered negative. This gives an accuracy of 0.80, which is much better than the ratings of the current competitors.

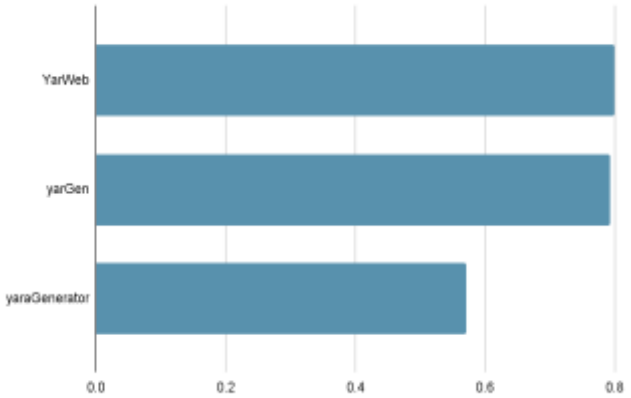

Fig. 8. Graphical representation of the accuracy of the rules produced by YarWeb and its competitors

SJIF Impact Factor (2024): 8.675| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 9 | Issue: 2 | February 2024 - Peer Reviewed Journal

VII. FUTURE WORK

The future for YarWeb is bright as the ability for the product to extend its capabilities is endless. For starters, the entire front end can be developed on React JS. In the realm of crafting immersive and responsive user interfaces, React JS stands as a compelling force. React encourages the creation of self-contained, reusable UI components. This modular structure promotes code organization, maintainability, and testability. Components can be developed and tested independently, fostering collaboration and streamlining workflows. React employs a virtual DOM, a lightweight in-memory representation of the UI. This enables efficient updates by pinpointing and rendering only the necessary changes, leading to faster and smoother user experiences. To enhance the encryption standards for the login process, sturdier encryption algorithms like SHA-256 can be used. This ensures that the user information is well protected and safeguarded from threat actors. Instead of storing user data in a simple .db file, a node JS setup can be used as a backend data server to store sensitive data. Furthermore, additional features can be added for the user/admin that enhance the user workflow.

When it comes to rule generation, using a bigger database with multiple datasets can be valuable. This would ensure that the rule is more accurate than before. however, this may or may not affect the loading speed. Using blockchain technology can help secure and track the uploaded files better since it works on a decentralized platform. Moreover, the constant updation of databases would help to address and target emerging malware threats and signatures making YarWeb more efficient than it already is.

VIII. CONCLUSION

YarWeb is created with simplistic user interaction and ease of usage in mind. The unique selling point (USP) of the product is its portability and production-ready environment. The team behind YarWeb has analyzed almost all the tools related to YARA rule generation, collected and recorded the drawbacks, and aided in creating YarWeb to combat these issues. Currently, YarWeb can produce YARA rules in under 5 seconds, and does it entirely on a webpage with an accuracy of 80%. Malicious signature is a topic many engineers and developers sideline due to its complex nature. YarWeb makes learning and using YARA rules fun and intuitive, opening doors to the world of malware analysis and beyond.

AVAILABILITY

The YarWeb reference implementation can be obtained at https://github.com/issashrez/YarWeb.

ACKNOWLEDGEMENTS

We would like to thank VirusTotal for providing the API key and helping us and our users scan their files. We would also like to express our gratitude to MalwareBazaar for providing us with all the necessary malware samples to aid us both in database creation and for testing the samples against our YARA rules which helped us to track the precision and accuracy of the model.

REFERENCES

1. *Khalid, M., Ismail, M., Hussain, M., and Hanif Durad, M. (2020). "Automatic YARA Rule Generation", 2020 International Conference on Cyber Warfare and Security (ICCWS). doi:10.1109/iccws48432.2020.92923*
2. *E. Raff, R. Zak, G. L. Munoz, W. Fleming, H. S. Anderson, B. Filar, C. Nicholas, and J. Holt, "Automatic Yara Rule Generation Using Biclustering", https://doi.org/10.48550/arXiv.2009.03779*
3. *N. Naik, P. Jenkins, R. Cooke, J. Gillett, and Y. Jin, "Evaluating automatically generated YARA rules and enhancing their effectiveness," in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), Dec. 2020, pp. 1146–1153.*
4. *N. Naik, P. Jenkins, N. Savage, L. Yang, K. Naik, and J. Song, "Embedding fuzzy rules with YARA rules for performance optimization of malware analysis," in IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2020.*
5. *VirusTotal. (2019) YARA in a nutshell. [Online]. Available: https://virustotal.github.io/yara/*
6. *V. Alvarez. (2019) Writing YARA rules [Online]. Available: https://yara.readthedocs.io/en/v3.4.0/writingrules.html*
7. *Readthedocs. (2019) Writing YARA rules. [Online]. Available: https://yara.readthedocs.io/en/v3.5.0/writingrules.html*
8. *C. Doman. (2018) yabin: A YARA rule generator for finding related samples and hunting. [Online]. Available: https://github.com/AlienVault-OTX/yabin*
9. *C. Clark. (2013) yaraGenerator: Automatic YARA rule generation. [Online]. Available: https://github.com/Xen0ph0n/YaraGenerator*
10. *F. Roth. (2018) yarGen is a generator for YARA rules. [Online].Available: https://github.com/Neo23x0/yarGen*