**RANSOMWARE DETECTION IN A SYSTEM**

**[1]Y Tejaswi, [2]V Guna Sri, [3]V Divya Siva Ganga, [4] P Madhu Hasitha**Shri Vishnu Engineering College for Women and Information Technology Bhimavaram, West Godavari, India.

20b01a12j3@svecw.edu.in, 20b01a12h5@svecw.edu.in, 20b01a12i9@svecw.edu.in, 20b01a12d8@svecw.edu.in

***Abstract - In this research paper, we present a proactive approach to ransomware detection using machine learning techniques. Ransomware continues to pose a significant threat to cybersecurity, exploiting vulnerabilities in systems and encrypting critical data for extortion. Traditional signature-based methods often struggle to keep pace with the rapid evolution of ransomware variants. Therefore, we propose a comprehensive framework that encompasses data collection, feature engineering, model selection, evaluation, and deployment. Through extensive experimentation and evaluation, we demonstrate the effectiveness of our approach in detecting ransomware behaviour while minimizing false positives. Key facilities and tools for ransomware analysis are also presented along with open challenges. This paper proposes a ransomware detection method that can distinguish between ransomware and benign files as well as between ransomware and malware. The experimental results show that our proposed method can detect ransomware among malware and benign files.***

**I. INTRODUCTION**

Ransomware is a malicious software that encrypts a user's files or locks the system, demanding payment for their release. Traditional signature-based methods may struggle to keep pace with evolving ransomware variants. Machine learning offers a proactive approach by learning patterns indicative of ransomware behaviour, aiding in early detection and prevention. The proliferation of ransomware attacks has underscored the need for proactive measures to detect and mitigate these threats. Machine learning, with its ability to learn patterns from data, offers a promising avenue for early ransomware detection. Traditional ransomware detection techniques including event-based, statistical-based, and data-centric-based techniques are not adequate to combat. This research paper seeks to address the critical challenge of combating malware and ransomware through a proactive approach grounded in machine learning techniques. By leveraging the power of data-driven algorithms to analyse patterns and anomalies in system behaviour, we aim to develop a robust framework for detecting and mitigating these malicious threats. Our research will contribute to the broader cybersecurity community by offering insights, methodologies, and best practices for enhancing the resilience of organizations against the ever-evolving landscape of cyber threats.



**II. URL - BASED DETECTION**

In our project we implemented URL based detection. Malware URLs have been compromised URLs that are used in intrusions. As demonstrated by the fact that about one-third of all websites have a risk of being hazardous, malicious URLs tend to be employed in committing cybercrimes [106]. In order to trigger attacks, a fake or hostile website hosts a range of unwanted information in the form of emails, phishing messages, or download drives. When uninformed consumers visit these websites, they can fall victim to an array of frauds, such as theft of identities, credit card theft, and malware installation, in addition to monetary harm. Spam, phishing and manipulation of emotions, download drives, and other popular methods of attack utilizing URLs that are malicious are all prevalent. Drive-by download describes the unintentional download of malware during an ordinary website visiting a URL.

These types of assaults occur frequently, resulted in billions of dollars in losses, and some even took benefit from catastrophic events [182]. Algorithms that are adept at quickly recognizing these detrimental URLs can be very helpful in fighting an extensive variety of cyber-security problems. As a consequence, professionals and academics have worked to come up with effective treatments regarding malicious URL identification.The blacklist technique is an especially common approach used across different malware groups for the recognizing of the bad URLs.

In simple terms, blacklists are databases of URLs that have been independently certified to be malicious in the past. This database is constructed up progressively occasionally using crowdsourced tools like PhishTank, whenever it becomes known that a certain URL is malicious.

However, maintaining up to date with an exhaustive list of malicious URLs is not feasible, especially when new URLs are created every day. Attackers utilize sophisticated techniques to get over blacklists and deceive consumers by obfuscating the URL to make it "look" authentic. Using an Internet Protocol (IP) address, another domain, long host names, and typos are some methods of hiding the host. By modifying the malicious URL, all of these websites seek to hide the website's malicious intentions. The practice of hiding a malicious URL behind an URL that is brief is referred to as obfuscation, and it has gained prominence recently due to URL reduction services. Users access the URLs once they seem genuine, at the moment when an attack starts.

This is often performed by malicious code incorporated into the JavaScript. Often attackers will attempt to obfuscate the code consequently as to avoid that it has been signed centred tools from detecting them. Attackers use many other techniques to avoid blacklists such as: fast-flux, in which proxies are immediately generated to host the webpage; computational subsequent generations of new URLs; etc. Additionally, attackers can concurrently begin more than one attack to change the attack-signature, making it indistinguishable by tools that concentrate on unique signatures from others. Blacklisting methods, thus have severe constraints, and it shows up almost insignificant to skip them, particularly because blacklists are useless for establishing predictions on new URLs. Subsequently we provide an in-depth review of the several feature representation techniques applied to this topic. Following that, other algorithms that were built based on the features of URL data and used to solve the problem are shown.

Finally, we go over the recently introduced idea of malicious URL detection as a service and the rules that need to follow when designing a system of that type.

**III*.* OVERVIEW OF PRINCIPLES OF DETECTIN MALICIOUS URLS**

Several approaches have been tried to address the issue of malicious URL detection. We developed our project using Machine Learning techniques. These techniques try to analyse the data that comprises a URL and the websites or webpages that correspond to it by obtaining precise representations of features of the URL while employing training data gathered from safe and malicious URLs to develop a model for prediction. Features can be used in two different ways: as static features or as dynamic features. While conducting static analysis, we analyse a webpage utilizing information that is readily accessible without actually accessing the URL. The features that get extracted comprise host information, lexical features from the URL string, sometimes even HTML and JavaScript content. These approaches are safer over the dynamic ones as they do not require implementation. Beneath this is a belief that safe and malicious URLs distribute certain features in different ways. Static methods of analysis have been extensively studied by applying machine learning techniques because of the relatively secure environment for retrieving essential information and the ability to adapt to all the types of the risks.

In this study, we concentrate on static analysis methods, which have experienced great success with machine learning. One of the techniques used in dynamic analysis involves maintaining an eye on the actions of systems that might become victims and seeking out any unusual behaviors. These are composed of systems which analyse internet access logs for unusual activities and analyse system call patterns for unusual behaviors. Techniques for dynamic analysis are complex to apply and generalize, and as such contain inherent risks.

**IV*.* LITERATURE REVIEW**

An outline of the behavior of the process is made up of many actions, like file management, which are made up of several operations. When we record process behavior as a series of API calls, each API call represents a process operation, each sequence of API calls represents an activity, and the entire recorded sequence of API calls represents a process behavior. The structure of texts is comparable to this hierarchical design. One of the compositions is made up of several sentences that each contain several words. Because of this, we assume that we can extract the feature of process behavior by using language model and we can train the feature of process behaviour to RNN. After training, we extract features as feature vectors from the process for validation using trained RNN. An image is created using the feature vectors that were taken out of the process behavior log. There could be a number of local elements in this picture that correspond to process operations. By training local characteristics, we use CNN to classify these photos. To build a classifier, CNN images both benign and malicious processes. Our idea uses appropriate DNN and splits process classification into two portions. There are four sections to the training flow as First, log files are created and process activity is observed. Second, log files are used to train the RNN to create a behavioral language model. Third, a trained RNN extracts features from the log file and transforms them into a feature image. Finally, the CNN is trained with training feature images which has malware or benign label.

 **V.FEATURE EXTRACTION**

As mentioned earlier, the accuracy of the training data, which depends on the quality of feature representation, is an important aspect when evaluating the performance of the machine learning model.

The following two phases can be further subdivided in the feature representation process:

**1. Feature Collection:** The objective of this engineering-focused phase is to gather relevant information about the URL. This includes characteristics such if the URLs are on a blacklist, characteristics derived from the URL string, host information, website content (such as HTML and JavaScript), popularity data, etc.

**2. Feature Processing:** The second step is feature preprocessing, which involves properly formatting and turning unstructured information about the URL (such a written description) into a vector of numbers so that machine learning algorithms can use it.

In some cases, the numerical data can be used just as it is, while textual or lexical material is frequently conveyed using bags-of-words. Experts have set forth several feature classifications for malicious URL detection that can be employed to provide useful data. These features can be broken down into four categories: content-driven features, host-driven features, lexical features based on URLs, blacklisted features, and others (popularity and context). Each has benefits as well as drawbacks; some are extremely informative, but acquiring these features can be highly costly. Similar preprocessing challenges and safety risks differ among features. We are going to continue into great depth about each of these feature categories. For a URL, a lot of various kinds of data are available. It requires an extensive amount of resources to search for the data and transform the data that is unstructured into a vector of features which is compatible with machine learning. While additional data can enhance models for prediction (depending on obtaining a large number of features can often be impractical (because of the inadequate regularization). For example, a number of host-based features could take just a few seconds to load, thereby rendering employing them in an actual environment unfeasible. The material and its features carry the greatest risk in terms of related security risks since it is feasible for malicious software to be purposefully downloaded while trying to use these features; other features are not impacted by the above issues. If the entire blacklist can be maintained locally, the collection overhead is negligible. However, if the third-party dependency requires to be accessed during runtime, the collection time of the blacklist features may be lengthy. Since the lexical features are basically direct extensions of the URL string, their collection is quite efficient. Acquiring host-based features takes a bit of while. The time frame of feature collecting is impacted by content features, which typically necessitate downloading the webpage. After the information has been collected, preprocessing often involves computing rapidly feature extraction. Lexical characteristics, like unstructured data, have a very high dimensionality in terms of size. attributes of the hosts and the content). The main reason for this is that they are all kept as Bag-of-Words characteristics.

**Fig.** Early fusion strategy



**Fig.** Late fusion strategy



**Fig.** Intermediate fusion strategy

**VI*.* MACHINE LEARNING ALGORITHM FOR MALICIOUS URL DETECTION**

To categorize URLs as dangerous or benign, train a machine learning model.

For training, use a dataset of labelled URLs. You can use well-known libraries like scikit-learn. And consider the algorithms such as AdaBoost Classifier and Random Forest. Here in our project we used the above two algorithms for detecting the URL based ransomware in a system.

***1. RANDOM FOREST***

Random Forest is an ensemble learning algorithm that is widely used for both classification and regression tasks. It is based on the concept of decision trees, where multiple decision trees are trained and their outputs are combined to make a more accurate and robust prediction. In the context of the URL Threat Detection System project, Random Forest can be used for classifying URLs into benign or malicious categories.

*Utility in the Project for a URL Threat Detection System:*

1. **Managing Variable Features:** URLs can have a number of characteristics, including the age of the domain, length, HTTPS presence, and more. Many different features can be handled by Random Forest in an efficient manner.
2. **Strongness and Broadness:** Random Forest is more resilient to noise and outliers in the data because of its ensemble nature. It is appropriate for real-world applications because it generalizes well to unknown URLs.
3. **The significance of features:** A feature importance metric is provided by Random Forest, which shows how important each feature is for predicting outcomes. Understanding which elements are most important in recognizing malicious URLs can be done with the use of this information.
4. **How to Avoid Overfitting:** In order to avoid overfitting and ensure that the model does not memorize the data, randomization and the usage of several trees.

In summary, Random Forest is a powerful and versatile algorithm that can be beneficial in the URL Threat Detection System project for its ability to handle diverse features, reduce overfitting, and provide robust predictions.

***2. AdaBOOST CLASSIFIER***

AdaBoost is an ensemble learning technique that builds a strong classifier by combining several weak classifiers. AdaBoost could be used to aggregate the predictions of several weak classifiers, each trained on a portion of features or a subset of the dataset, in the context of URL-based detection. AdaBoost trains weak classifiers successively on various subsets of the training data, giving more weight to the samples that the preceding classifiers incorrectly classified. AdaBoost highlights difficult samples by iteratively modifying the weights of misclassified samples, which enhances the performance of the model as a whole.

*Utility in the Project for a URL Threat Detection System:*

1. **Improved Detection Accuracy:** AdaBoost's capacity to turn a number of weak classifiers into one powerful classifier can improve the system's overall detection accuracy. AdaBoost improves the system's capacity to distinguish between benign and harmful URLs by iteratively modifying the weights of misclassified instances. This allows AdaBoost to concentrate on threats that are challenging to identify, such malware URLs.
2. **Handling Imbalanced Data:** In real-world situations, there are frequently significantly more benign URLs than harmful URLs, which leads to unbalanced statistics. AdaBoost can handle imbalanced data well because of its training method, which gives larger weights to instances that are misclassified and gives more weight to the minority class (malicious URLs). This enhances the system's capacity to identify uncommon risks and helps keep the model from being skewed in favor of the majority class.
3. **Robustness to Overfitting:** AdaBoost is less prone to overfitting compared to some other complex machine learning algorithms. By combining multiple weak classifiers and focusing on the most challenging instances, AdaBoost helps prevent the model from memorizing noise in the training data and captures underlying patterns more effectively. This results in a more robust and generalizable detection system that performs well on unseen data.

Overall, AdaBoost classifier can significantly enhance the performance, robustness, and interpretability of a URL threat detection system. By leveraging its strengths in handling imbalanced data, preventing overfitting, and combining weak classifiers, AdaBoost contributes to building a more effective and reliable defense against malicious URLs and cyber threats.

**VII. RESULTS**









**VIII*.* DESIGN PRINCIPLES**

Delivering a variety of goals is our aim when building a machine learning-based harmful URLs detection system in real-world applications. A wide range of goals and requirements are

To achieve the goals that were set, choices must be made. In the section that follows further down, we go over the key aspects to consider about in in simple terms:

1. **Accuracy:** One of the most significant goals for any malicious urls detection is often to attain accuracy. The most ideal scenario would be to minimize false positives—the incorrect detection of harmless URLs that are categorized as malicious—while increasing the identification of all risks that presented by malicious URLs ("true positives"). A realistic malicious URL detection system needs to frequently trade off among the ratios of positive results and negative results by changing detection thresholds based on the requirements of an application  because no method can guarantee perfect accuracy in detection.
2. **Detection Speed:** For an effective malicious URLs detection system, speed of detection is essential especially for online platforms or applications related to cybersecurity. In the case that a user posts a fresh URL on a social media site like Instagram, for instance, an ideal system should be able recognize the malicious URL immediately and block the URL and associated posts in real time in order to stop threats and damage to the public. Some applications related to cybersecurity might have stricter requirements for detection speed; for example, detection might require taking place in nanoseconds so that a malicious URL request can be stopped immediately and in real time if a user opens on that.
3. **Scalability:** A malicious URL detecting system in the real world need to be capable to scale up in order to train its models with millions or billions of training data, given the enormous number of URLs that are always increasing. Two primary groups of tasks and approaches are needed for achieving the high scalability required. Researching methods of learning that become more scalable and effective comes initially.
4. **Adaptations:** In action, malicious URLs detection mechanisms have to deal with many different practical difficulties, such as competitive structure like concept drifting, where the distribution of malicious URLs develops over time or happens in an adversarial manner to evade the detection system, missing values (such as unavailable features or features which are too costly to compute in a promptly manner), a growing amount of new characteristics, etc.

To operate effectively and consistently in a broad range of situations, an URL that is malicious detection system in the real-world setting needs to have an important ability for flexibility.

1. **Flexibility:** A practical ML-based risky URL detection tool should be built with sufficient adaptability to allow for easy improvements and modifications, given the complex nature associated with dangerous URL detection. Amongst them is the immediate updating of the model predictions in the relation to being flexible to be expanded for the training models to deal with a variety of new attacks and dangers, getting simple to swap out the algorithm used for training and models whenever there is a need, getting responsive to new data used for training, and, at last, having the ability to interact with people wherever there is a need, such as active learning for performance improvement.

**IX. TOOLS AND SERVICES FOR RANSOMWARE ANALYSIS**

 Related research has utilized the use of a broad spectrum of computer software resources and features due to the wide spectrum of technological tasks involved in a ransomware investigation. Here is a brief overview of these:

1. *Malicious File Stores:*

Several academics have assembled distinct datasets of ransomware on their own behalf. Some have, however, made use of resources like Virus Total and the VirusShare website, which house sizable malware repositories with the numerous ransomware families. Consumers generally give these malicious binaries, which may then be downloaded for an in-depth look.

1. *Raw Trace Capture of:*

To inspect ransomware binaries and gather trace files, sandbox and virtualization methods are often utilized. The best-known solutions for Windows-based testing include the Cuckoo sandbox as well as the Triage facility (www.tri.ge), which in addition has a robust online sandbox. In particular, researchers can get pre-loaded/pre-processed reports and malware binaries, and they can also upload and execute malware binaries, utilizing the latter resource.

1. *Pre-Processing/Feature Extraction method:*

For selecting and extract training features, machine learning algorithms necessitate a thorough pre-processing of the information. For these explanations, a lot of researchers used to write their own specifically customized programs in languages like C/C++, Python, Java, etcetera. But most ML packages (shown in greater detail below) already contain a wealth of abilities to facilitate this kind of processing. In addition, the open-source Pandas library for the Java programming language includes powerful data translation and manipulation features for datasets with labelling.

1. *Machine Learning:*

A broad spectrum of open-source and free machine learning (ML) applications are currently reachable. Amongst these are toolkits like Scikit, TensorFlow, PyTorch, and Weka, and Keras.io. Together collectively, these solutions provide complete assistance for nearly all supervised as well as unsupervised machine learning algorithms, including decision tree models, k-NN, k means clustering, RF, the SVM, and a majority of NN-based editions.

**X. OPEN CHALLENGES**

Long into the future, the threat of ransomware will continue to be a concern. Consequently, it is essential to stay up current with these shifting circumstances and supply effective structures for detection and classification. All in all, it is obvious that the works examined here represent a remarkable collection of contributions in this field. However, there are still a lot of unresolved problems which must be addressed with; some of these are briefly mentioned here.

Firstly, the test case scenario and ransom datasets require immediate attention to be standardised. Comparing current techniques is really challenging because most researchers have used their own datasets to assess distinct ransomware subsets. As an outcome, it's essential to select a small number of the most crucial ransomware families and create/maintain a binary downloadable repository for each. likewise it needs to be done to define related testing and performance variables, such as measurement metrics, sandbox or virtual machines run-times, etc. These general steps will facilitate accurate of comparative analysis and aid to increasing of the reproducibility.
In addition, for Machine learning -based detection and categorization, it is the critical to solve increasing security and scalability issues. Particularly, despite the fact that many thoughts have been put up, their practicality in real-life scenarios has not been fully investigated. In particular, a lot of consumers could be reluctant to open up their extensive log files, or traces, for outside examination. On the other hand, local host processor can find it too difficult to handle full local pre-processing of raw data. Furthermore, executing centralized data gathering and predictive processing at just one location can become difficult. Therefore, it's even more important for developing suitable structures to deal all these issues.

**XI. REFERENCES**

 [1] Cisco System, "Ransomware defence validated architecture guide," 2016.

 [2] Safety Investigators "Ransomware facts, tendencies statistics of 2022," 2022.

 [3] A. Kapoor, "An overview and planned developments of ransomware detection, avoiding problems, and prevention scheme," Sustainable development, vol. 14, no. 1, Dec. 2021.

 [4] "Senate bill will mandate ransomware payment reporting and attacks on computers," The government of Bloomberg, 28 September 2021.

 [5] A survey on detecting strategies for cryptographic ransomware, E. Berrueta, D. Morato, E. Magana, and M. Izal, Access, IEEE, vol. 7, pp. 144925–144944, October 2019.

 [6] A survey on Windows systems ransomware taxonomy and detection techniques, R. Moussaileb, N. Cuppens, J.-L. Lanet, and Bouder, ACM Computational Surveys, vol. 54, no. 6, 2022, July.

[7] Ahmet Cosar, Betul Altay, & Tansel Dokeroglu. the year 2018. Techniques for detecting illegal websites with trained machine learning that depend on the density of keywords and context. Soft computing was introduced in 2018.

 [8] Joel Ruben, Kshitij Gorde, and Ankesh Anand Tanmoy Chakraborty, Anthony Moniz, Noseong Park, & Bei-Tseng Chu.

 the year 2018. Applying text generative neural networks, phishing URL detecting with oversampling is achieved. The use of big data is the topic of the 2018 IEEE International Conference on. IEEE, the pages 1168–1177.

 [9] Nick Feamster, Wenke Lee, David Dagon, Roberto Perdisci, and Manos Antonakakis in 2010. Building a DNS Innovative Reliability System. at the security symposium of USENIX.

 [10] David Dagon, Wenke Lee, Roberto Perdisci, Manos Antonakakis, & Nikolaos Vasiloglou. 2011. Detecting Malicious Domains in the Greater DNS Level. during the security symposium of the USENIX Corporation.

[12] Kalyan Veeramachaneni, Ankit Arun, Sumeeth Kyathanahalli, and Ignacio Arnaldo. in 2018. Achieve, modify, and get ready: continuous learning to block harmful domains. The rise of big data is the topic of the 2018 international conference by the IEEE on the IEEE, 1891–1898.

 [13] In the year 2016, A. Piccolo, M. Gaudioso, A. Chiarello, and AAstorino. identifying malicious URLs using spherical classification. Benefits of Neural Computing (2016).

 [14] Sergio Villegas, Ivan Torroledo, Luis David Camacho, and Alejandro Correa Bahnsen in 2018. DeepPhish: AI Virus Simulator. In San Diego, California, United States, at the Conference on Electronic Crime Investigation, Proceedings, 15–17.

 [15] Lawrence K. Saul, Stefan Savage, and Sushma Nagesh Bannur. 2011. Analyzing a website based on its content: identifying the linguistic, related to structure, and aesthetic features that distinguish dangerous website. The 4th ACM Workshop on Privacy and Artificial Intelligent Systems Contents.