IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 February 2025, accepted 20 March 2025, date of publication 26 March 2025, date of current version 4 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3555163

RESEARCH ARTICLE

RanViz: Ransomware Visualization and Classification Based on Time-Series Categorical Representation of API Calls

VHUHWAVHO MOKOMA AND AVINASH SINGH
Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa
Corresponding author: Vhuhwavho Mokoma (u20470992@tuks.co.za)

ABSTRACT Ransomware continues to pose a significant threat to individuals and organizations worldwide, causing disruptions, financial losses, and reputational damage. As ransomware attacks grow in sophistication, understanding their behaviour through effective analysis has become increasingly critical for mitigation and prevention. However, ransomware analysis presents several challenges. First, the sheer volume of Application Programming Interface (API) call data generated by ransomware during execution can overwhelm traditional analysis methods. Second, the temporal and categorical nature of this data makes identifying meaningful patterns complex. Third, the integration of machine learning (ML) models, which are essential for accurate classification, is hindered by the difficulty of modelling intricate API call behaviours. Without effective tools to address these issues, analysts risk missing critical behavioural indicators. To overcome these challenges, the proposed Ransomware Visualization (RanViz) system was developed to provide a comprehensive visual analytics and classification platform designed to enhance ransomware analysis. RanViz employs advanced visualization techniques to represent categorical API call time-series data, enabling analysts to intuitively understand ransomware behaviours that might otherwise remain obscured. The system incorporates ML models based on API call frequency, temporal interval, and sequence to classify unknown samples as either benign or ransomware. The models collectively achieve an accuracy of over 95% in detecting ransomware. By providing a unified platform that combines powerful visualization tools with high-performing ML models, RanViz simplifies ransomware analysis and offers a robust framework for accurate classification. This makes it an invaluable tool for digital forensics and cybersecurity professionals tasked with addressing the ever-evolving ransomware threat.

INDEX TERMS Ransomware analysis, API calls, machine learning, ransomware, time series, visualization.

I. INTRODUCTION

Ransomware is a type of malware that encrypts files or blocks system access, demanding a ransom from the victim to restore functionality [1], [2]. The victim can also be threatened by the public release or deletion of sensitive files [1]. New techniques to launch advanced attacks and avoid detection have been developed in the evolution of ransomware [1]. The cybersecurity firm Bromium has reported that there has been an increase of 600% in the number of ransomware families from 2013 [3]. Ransomware continues to be a major threat to organizations across the world [4], [5], [6], [7]. Recently, Insomniac Games suffered a ransomware attack that led to the public release of over 1.3 million files with the personal data of employees and information about video games that are still in development [8]. In 2015, attackers gained $325 million in a single ransomware attack [3]. The financial and operational costs of ransomware attacks negatively impact organizations and therefore ransomware cannot be ignored.

One of the methods to deal with ransomware is ransomware analysis. This involves the use of static, dynamic, or hybrid analysis techniques. Static analysis is the analysis

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

of malware without executing the sample. Dynamic analysis is the analysis of the behaviour of malware that has been executed within a controlled environment. Hybrid analysis involves the incorporation of both static analysis and dynamic analysis in the analysis process [9]. There has been a shift towards behaviour-based ransomware detection methods. However, it is still difficult to accurately distinguish between benign and malicious behaviour [10]. The development of techniques used for ransomware detection and analysis is a never-ending process to counter the evolving nature of ransomware [11]. Analysis is crucial in the building of defence and detection systems against any type of malware [12]. It is also useful for the post-incident analysis of malware attacks for digital forensics purposes. Application Programming Interface (API) calls are useful in helping to understand the behaviour of ransomware and form part of dynamic analysis [13]. The relevant feature classes in the dynamic analysis of ransomware are the registry key and API calls [14]. The relevance of API calls is because API calls facilitate the communication between various components of the computer system by being the interface between software components in the computer system. Analysts can gain valuable insights into the behaviour of ransomware from API calls. By examining the sequence and frequency of API calls, analysts can learn aspects of ransomware such as command-and-control interactions, file encryption techniques, and transmission techniques.

Visualization systems can be used to provide effective analysis of ransomware behaviour. Visualization systems simplify the analysis process by creating graphical data representations [15]. There has been work done to create visualization systems such as MalView [12], Knowledge Assisted Visual Malware Analysis System (KAMAS) [16], and Eventpad [17]. Malware visualization systems have two main objectives: To help analysts gain insights into the behaviour of a malicious sample and to recognize shared behaviours across malware samples. To achieve these objectives, three types of malware visualizations exist: malware analysis, malware comparison and malware summarization [12], [18]. Malware analysis refers to the analysis of an individual malware sample by visualizing it to uncover its behaviour and reconstruct the sequence of malicious events. The sample is studied to uncover its behaviour that is relevant to malware forensics. Malware comparison refers to the comparison of multiple malware samples to uncover the shared behaviour among them. This helps the analyst to uncover similar variants and families of ransomware based on their behaviour. Malware summarization refers to the visualization of the summarized behaviour of a related group of samples [18]. This is distinct from malware comparison because it is a visual summary generated by the visualization system and not a comparison between multiple samples conducted by the analyst.

Although there are multiple existing malware visualization systems and analysis techniques, analysts continue to handle many complex samples of ransomware with difficulties in analyzing raw API call data [16]. Manual analysis of API calls is time-consuming because of the large volume of API calls performed during the execution of ransomware and the complexity of modern software systems. The categorical and temporal aspects of API call behaviour are difficult to uncover using typical analysis techniques. This reduces the ability to recognize important patterns and anomalies linked to ransomware behaviour.

II. PROBLEM STATEMENT

Digital forensics and cybersecurity analysts face a problem during the manual analysis of ransomware because of its complexity. It is usually time-consuming and ineffective. Existing visual analytics systems limit the ability of analysts to uncover patterns and anomalies because analysts still have to manually examine all the visualized data. Effective analysis is also limited by the lack of analysis focused on API calls and the application of machine learning to visual analytics systems. To effectively analyze ransomware behaviour, a visual analytics system that uses a time-series categorical representation of API calls and applies machine learning must be developed. A time-series categorical representation of API calls allows categorical and temporal behaviours or anomalies to be visualized for analysts to easily see by providing clear insights. The Ransomware Visualization (RanViz) system classifies the samples provided by analysts using a machine learning model based on API call behaviour. RanViz reduces the complexity of ransomware analysis and improves its effectiveness. The contributions of this research are as follows:

- Building an analysis system for ransomware that uses the time-series categorical representation of API calls.
- Building an analysis system that contains all aspects of malware visualization such as malware analysis, comparison, and summarization.
- Training machine learning models based on the temporal and categorical behaviours of API calls and applying machine learning to the system for ransomware classification.
- Creating new ransomware visualizations, utilizing existing visualization and encoding techniques to improve and simplify the analysis.

III. BACKGROUND

The evolution of ransomware has a foundation in the technological advancements of the modern computing era. The development of the Internet and the rapid increase in the use of devices such as mobile phones, laptops, servers, and computers have created a network of potential targets for ransomware attacks [1]. The development of encryption techniques, namely the RSA and AES algorithms for security purposes, has also been misused by attackers to implement sophisticated ransomware encryption that is difficult to crack. The development of cryptocurrencies has also been misused to allow attackers to receive payment from victims in a way that hides their activities from detection [1], [19]. There

are 30 identified families of ransomware that use Bitcoin cryptocurrency to source payment from victims [1]. The increasing trend of using a ransomware-as-a-service (RaaS) which provides tools and infrastructure to cybercriminals has made it easier for novice criminals to execute ransomware attacks. This has resulted in an increase in the number of ransomware attacks [20], [21]. It is important that ransomware detection techniques are able to detect RaaS. The development of organizational servers and the increase in the use of cloud services by organizations to store massive amounts of sensitive data have created valuable targets for ransomware attacks. All of the previously stated advancements have provided a solid foundation for the evolution of ransomware.

Visualization tools have the main intention of being an effective means of communication by providing visual trends, patterns, structures and anomalies. The purpose of visualization is to transform raw unstructured data into information that can be easily understood using different shapes, colours and dimensions [22]. A common limitation of visualization is when the size of the input increases and affects the performance of the visualization. This can also result from data that has not been processed correctly [23]. There are six types of visualization: Temporal Visualization, Geo-Spatial Visualization, Hierarchical Visualization, Network Visualization, Multidimensional Visualization and Field Visualization [24]. Temporal Visualization is based on time-oriented data and has a time dimension. Geo-spatial Visualization is based on geo-related data and have applications in carthography. Hierarchical Visualization is based on aggregated data on multiple levels. Network Visualization is based on datapoint connections. The data points can be multidimensional. Multidimensional Visualization is based on data with multiple dimensions including 2D and 3D. Field Visualization is based on rendering fields in 2D or 3D with geometric glyphs, flow glyphs, isolines, isosurfaces or streamlines [24].

There are two broad categories of ransomware: crypto-ransomware and locker ransomware [25]. Crypto ransomware is a category of ransomware that has the characteristic of encrypting the files or data of the victim [26]. Locker ransomware is a category of ransomware that has the characteristic of locking the entire system making it inaccessible to the victim [26]. These two ransomware categories are different, but they achieve the same goal of preventing the victim from accessing something important and enabling the extortion of the victim. After crypto ransomware infects the victim system it initiates a request to the command-and-control server and creates a connection to receive the encryption key that it needs to use [1]. After obtaining the encryption key, the ransomware searches for files containing sensitive data such as images, databases, documents, and spreadsheets [14]. The ransomware then uses an encryption algorithm to encrypt the files that it has found. A message then pops up to display the threat against the user, informing them that their files have been encrypted and that they have to pay a ransom to receive the key to decrypt the files [1]. In the case of locker ransomware, after it infects the system, it scans the system environment to determine the operating system information. It then modifies the operating system settings to restrict the user from accessing the system. Locker ransomware can also disable peripheral devices, such as the mouse and keyboard. In some cases, it may allow the user limited use of the system so that they can make the ransom payment [26]. It can also disable system processes to prevent the user from terminating malicious processes to regain access to the system. It does not encrypt data stored in the system. Data can still be recovered if stored on a removable data drive [27]. A message also pops up to inform the user that their system has been locked and to regain access to the system, they must pay a ransom [14]. It is important to understand how ransomware acts in its malicious operations to see whether analysis techniques can correctly identify some of these malicious actions described above. The next section discusses the related existing malware analysis techniques and systems.

IV. RELATED WORK

Several innovative malware analysis systems and techniques have been developed [12], [14], [16], [17], [28]. Research has been conducted to apply existing techniques and develop new techniques to uncover the behaviour of malware as a broad category and ransomware specifically. However, there should be continuous improvements in malware analysis to keep up with the rapidly evolving nature of ransomware and the complexity of analysis.

Research has been conducted on developing malware visual analytics systems that aim to help analysts in the malware analysis process [12], [16], [17]. Nguyen et al. [12], developed a system called MalView which is an interactive visualization system that uses pattern-matching techniques for hybrid analysis of malware. MalView focuses on the process activity, Dynamic Linked Libraries (DLL) calls, and domain names that interact with the sample. MalView has a time-series processes visualization, a library matrix for visualizing the DLL calls for each of the processes, and a network for visualizing the process dependencies. All of these visualizations are intended to help the analyst uncover malicious process signatures and abnormalities in the process frequency of a sample. MalView uses VirusTotal [29] to identify domain names that interacted with the sample to determine whether the domains are malicious or benign. The system uses external analysis from VirusTotal and IPstack [30] instead of internal analysis. This is a drawback that may reduce the effectiveness of the system against new malicious domains because the system is dependent on external systems to enhance their analysis techniques to handle new variants. MalView does not use a dataset to train a model to classify malicious samples or behaviour, which reduces its effectiveness as a malware analysis system. Cappers et al. [17], developed a system called Eventpad which is a visualization system that is focused on network

traffic. Eventpad is aimed at allowing analysts to select their attributes and patterns of interest. Their approach involved setting up a virtual network and capturing packets using tshark [31]. The captured network activity was converted into blocks using Eventpad. After the blocks were captured, an analyst was able to visually encode them and select the attributes of interest. It uses visual encoding by the analyst, as well as the matching of patterns, to uncover malicious behaviour and patterns. Eventpad does not make use of a network activity dataset to train a model to classify malicious samples or behaviour which limits its effectiveness as a malware analysis system. Wagner et al. [16], developed a system called KAMAS, which is a visualization system focused on malware behaviour analysis. KAMAS creates rules from analyzed behaviour that are utilized to determine if a malware sample has malicious or benign behaviour. KAMAS integrates the knowledge database by Dornhackl et al. [32], which is related to a malware behaviour schema. KAMAS uses a call explorer that provides information about API call occurrences. The analyst uses the call explorer to identify relevant calls based on their occurrence. KAMAS also has a rule explorer that visualizes the rules that are present in the sample and their occurrence in a bar chart and numbers format. A graphical summary was developed to simplify textual representations for analysts to easily comprehend. KAMAS does not utilize a dataset to classify samples based on their API call occurrence which reduces its effectiveness as a malware analysis system. KAMAS focuses on occurrence and does not offer categorical and temporal analyses to help the analyst comprehend the behaviour of the sample in depth.

Research has been conducted to explore the use of machine learning in the classification and detection of ransomware using various approaches [14], [28], [33], [34]. Machine Learning techniques are superior in detecting ransomware compared to traditional methods due to their adaptability and providing real-time responses [35]. Machine learning models can extract patterns that indicate malicious activity on their own. Sgandurra et al. [14], developed EldeRan, which focuses on using machine learning for ransomware dynamic analysis. Sgandurra et al. [14], also explored the limitation of dynamic analysis and suggested possible solutions to the limitation. This study involved the use of 582 ransomware samples from 11 families that were sourced from VirusShare [36]. The features with the most relevance were API statistics and registry key operations. EldeRan uses Regularized Logistic Regression (RLR). To evaluate the effectiveness of the model, it was compared to Naïve Bayes (NB) and Support Vector Machine (SVM). RLR was chosen as the model for EldeRan because of its ability to incorporate new training samples into the model without the need to retrain the model. The EldeRan model achieved an accuracy of 96.3% which is higher than the other machine learning models it was compared to. One of the dynamic analysis limitations explored in the research involves ransomware that masks itself from detection by waiting for a period of time before executing any malicious activity. They can also wait for the user to initiate an action before executing malicious activity. Sgandurra et al. [14], proposed a solution to this limitation, which involves using the known patterns of ransomware system calls for encryption to implement a detection technique or system. Hammadeh et al. [28], developed a method for detecting ransomware using machine learning. This method focuses on the use of YARA [37] rules including API function rules, cryptographic rules, extension rules, and keyword rules to detect cryptographic signatures. Hammadeh et al. [28], explored the Support Vector Machine (SVM) because of its ability to classify non-linearly separable data. Logistic Regression (LR) was explored because it is less prone to overfitting. K-Nearest Neighbours (KNN) was explored because of its ability to group unknown data into categories with common features, and Long Short-Term Memory (LSTM) model was explored because it can handle time series data and learn from previous data. The LSTM had the highest accuracy of 99.08% because of its ability to learn from previous information. The LSTM's ability to learn from previous information was appropriate in this case because the dataset had sequential bits.

Panaras et al. [38], developed a novel cooperative clustering framework which uses machine learning across multiple detection agents in a distributed network environment. System logs, network traffic data, and process activity were continuously monitored to extract relevant features that are indicative of ransomware behaviour. Principal component analysis (PCA) was used to reduce the complexity of the features. Historical data was also incorporated into the feature extraction process to enable the system to uncover long-term trends that it can use to detect a ransomware attack. The dataset to train the models was made up of generated datasets that simulate a ransomware attack and real-world historical ransomware datasets. The cooperative clustering was trained and compared with decision trees, SVM, and a random forest tree. The cooperative clustering had the highest accuracy of 96.8% and the lowest false positive rate of 2.7%. Rafapa et al. [20], developed a novel approach that uses random forest trees for ransomware detection. They collected ransomware samples from 2021 to have a dataset that contains recent and relevant ransomware variants. They used both static and dynamic analysis to extract features that are indicative of ransomware behaviour. Multiple random forests were trained on different subsets of the dataset and aggregated to improve detection. The aggregated random forest trees had an accuracy of 94.2 % and F1-Score of 0.94.

API calls and their relevance in uncovering malware behaviour have been studied [13], [39]. Zhang et al. [39], developed a ransomware early detection and defence system that utilizes API sequencing. API sequences were enhanced using Wasserstein Gans + Gradient Penalty (WGAN GP) [40], which is a generative adversarial network. The enhanced dataset was then used to train a machine-learning classifier.

The performance of all the trained machine learning models namely, the Random Forest (RF), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), and Naïve Bayes (NB), significantly improved after the dataset was augmented using WGAN-GP. The malicious API sequences uncovered by the machine learning models were mapped to public security knowledge bases. Alazab et al. [13], developed a method to extract API call features and utilize them for malware behaviour analysis. A Python [41] program was created to categorize malware behaviour based on the API calls. The program compared API calls in a sample with the Windows API from the Microsoft Developer Network. The results of this research led to the discovery of six main categories of malicious API call-based behaviour: get file information, searching files to infect, changing file attributes, copy/delete files, read/write files, and move files.

As mentioned earlier, MalView, KAMAS, and EventPad are visualization systems with different approaches and gaps relevant to the development of RanViz. The solution to ransomware that hides itself from detection by delaying activity as discussed by Sgandurra et al. [14], is linked to the features that RanViz provides, which incorporate system-level API calls and ransomware classification based on patterns and behaviours. The use of API function rules in the research by Hammadeh et al. [28], is relevant in feature engineering and training the model for integration with RanViz. There is a gap in applying machine learning models, such as those discussed previously, in a malware visualization system that is based on API calls. The insights of Zang et al. [39], on API sequences are relevant to RanViz in exploring the machine learning model to apply API call sequences as a feature of the dataset. The work of Alazab et al. [13], in uncovering the six main categories of suspicious behaviour based on API call features, is useful in helping RanViz correctly determine which API call behaviours belong to the six main categories and can be highlighted as malicious in RanViz. The next sections detail the development of RanViz.

V. RANVIZ SYSTEM OVERVIEW

RanViz aims to visualize ransomware effectively and simplify ransomware analysis. Its main functions are centered on malware analysis, malware comparison and malware summarization. A Malware Feature Engineering (MalFe) [42] report is used as input for the system. For each sample, it generates various visualizations: treemap, time series, heatmap, and 3D visualization. The treemap shows the frequency of the different categories and types of API calls that occur in the sample. The time series shows the occurrence of different API call categories over time. The heatmap shows the intensity of API calls at different timestamps. The 3D visualization shows the frequency of different API call categories over time in a 3D format. The system integrates machine learning models to classify uploaded samples as either ransomware or benign based on API call features.

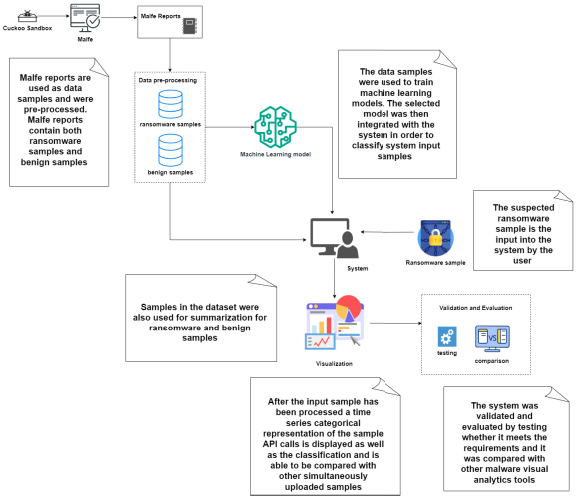


FIGURE 1. Technical system model.

The technical process model for the system is illustrated in Figure 1.

It uses MalFe reports as the dataset for machine learning model training and malware summarization. For malware summarization, the dataset is used to visualize the common behaviour of ransomware and benign samples. For malware analysis, the individual sample is visualized. For malware comparison multiple uploaded samples can be compared by the user. RanViz is integrated with the MalFe platform. To evaluate RanViz, it was compared with other existing malware visual analytics systems and through case studies. This will demonstrate the advances that RanViz brings in comparison with existing systems and its contributions to the broader field of ransomware analysis.

A. RanViz IMPLEMENTATION

RanViz was implemented using the Python Django web framework [43]. Python is suitable for RanViz because it has libraries that are compatible with machine learning and the processing of complex data. The visualizations were implemented using Apex Charts [44], an open-source JavaScript charting library. This was chosen because it provides interactive visualizations and allows the user to download the visualizations in png, svg, or csv form. This adds another layer of functionality to the RanViz system for record-keeping or for further analysis with other methods. Due to being integrated with MalFe, the security requirements and user authentication for the system are handled by MalFe. The machine learning models were integrated through the static folder used in Django in the joblib format. The next section discusses the RanViz process flow.

B. RanViz PROCESS FLOW

The system starts off with the user uploading a sample or multiple samples. If the user uploads multiple samples the system checks if the number of samples is within the limit

of nine and if they exceed the limit of nine the upload is denied. The purpose of allowing users to upload multiple files simultaneously is to address the problem of analysts dealing with large numbers of samples and improves the efficiency of the analysis process. If the uploaded sample is not a JSON file type or it does not have the correct executable format, the upload will also be denied. The system then checks whether the behaviour and process keys are present in the sample. The system then checks whether API calls are present. If they are present, it proceeds to process the sample for visualization and process the sample for submission to the integrated machine learning models. The uploaded report of a sample is preprocessed and converted into three python dictionaries based on the 280 API dict template that was used when training the models. The python dicts are based on the frequency of the API calls, the temporal interval of the API calls and the sequence of API calls. For frequency and temporal interval the logistic regression models are placed in the static folders of the system in a joblib format. To access the model in the system logic it is loaded from its file path in the static folder. The logistic regression model is used to determine whether an uploaded report is ransomware or benign based on the API call frequency. The dict values are converted into a numpy array that is fed as input to the loaded model. For API sequence the SVM model is placed in the static folder in joblib format. The outputs of the three models are processed using majority classification as the final result. After identifying whether the sample is ransomware or benign, the user can select the type of visualization they want. The user is then finally able to download the selected visualization. The process flow of the sample being uploaded until it is downloaded is shown in Figure 2. The next section describes each of the features of the system in more detail.

C. RanViz CLASSIFICATION AND ANALYSIS

RanViz uses simple language for the analyst. It hides the technical processes that are used by machine learning models and only informs the analysts of the output whether the sample is ransomware or benign. The successfully uploaded samples are processed in the integrated machine learning models and the classification output is displayed, as depicted in Figure 3. Each sample can be classified as either ransomware (red) or benign (green). This classification will help analysts classify new unknown samples or variants of ransomware. The initial display for each sample shows not only the classification but also the key statistical information of each sample, such as the number of processes and API calls. The user can compare this initial information among the uploaded samples through tab switching by selecting the tab button for the sample they want to display. The number of processes and API calls executed in a sample provides more information about the sample that may be useful for the user to understand the differences in the activity of ransomware versus benign samples. The next section discusses each of the visualizations provided by the system in detail and why they are useful.

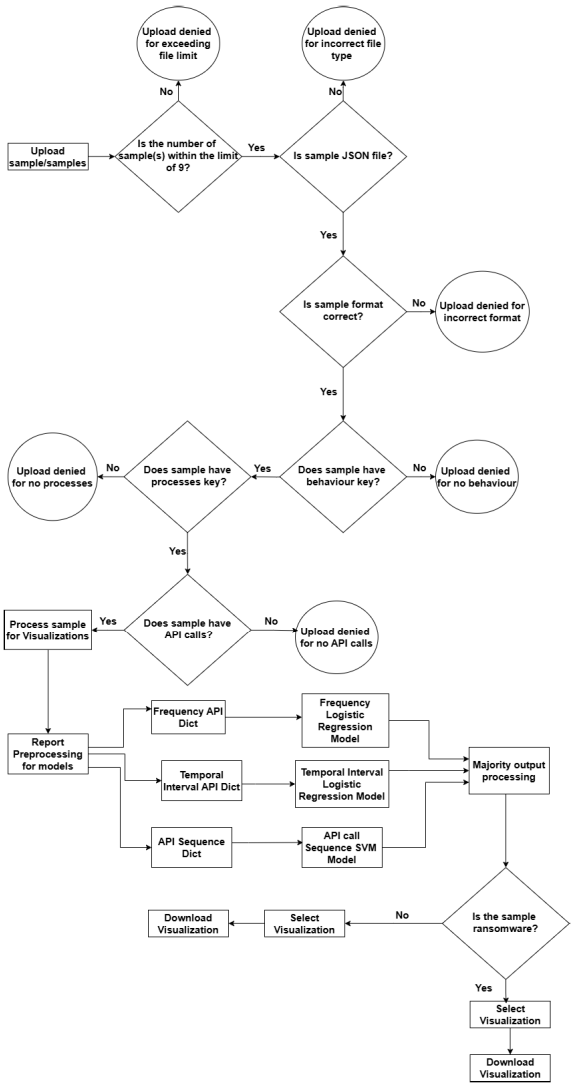


FIGURE 2. RanViz flow chart.

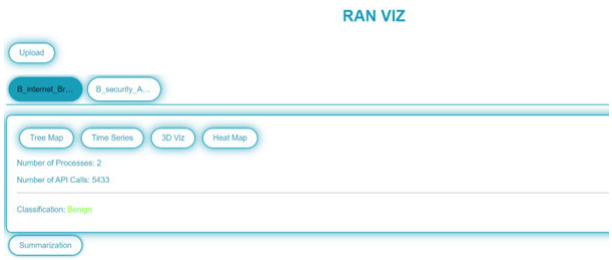


FIGURE 3. RanViz main page.

D. RanViz VISUALIZATIONS

RanViz uses a dashboard-style layout with all the visualizations clearly labelled within a section. The different visualizations give a more comprehensive understanding of the data for the analyst. The effectiveness of visualization depends on the quality of the data and its attributes [22]. In the case of RanViz, the data from the MalFe Reports provides attributes that are compatible with the visualizations

that are provided by RanViz with both categorical and temporal attributes.

For each of the uploaded samples, the user can select between four visualization options: treemap, time series, 3D visualization, and heatmap as depicted in Figure 3. The visualizations such as treemap, time series and heatmap are intuitive and do not require expertise in cybersecurity or machine learning. However, analysts will need to have knowledge about API calls to get an in depth understanding of what the visualizations communicates. The user can then compare the uploaded samples based on their selected visualization and switch between the uploaded samples. This allows the differences in behaviours between samples to be identified by the user. The details of the visualizations are further discussed in the sections that follow.

1) TREEMAP

The treemap visualization allows the user to see the relative frequency of the higher-level categories of API calls due to the frequency corresponding to the size of the API call category, as depicted in Figure 4. The treemap uses consistent colours for the API categories and types which improves the usability of the system. The treemap displays the data in a hierarchical structure which creates a different level of abstraction for each level. This helps the analyst to discover interdependencies between API call categories and types. By selecting a specific API call category, the user can drill down and see a treemap depicting the API call types within the selected API call category. These interactions and adaptability ensure that the analyst can easily understand the information presented through the visualization. In terms of visual scalability, the treemap is not affected by the increase in size of the input. The individual analysis of a sample allows the user to see which categories of API calls are the most prevalent within an individual sample. For example, a high prevalence of the GetFileInformationByHandle API call type indicates to the analyst that the sample conducted many operations to retrieve information about files in the system, indicating malicious behaviour before the files were encrypted by the sample. Such information allows the analyst to trace all malicious activities executed by the ransomware sample during its operation. For sample comparison, the user is can see the differences in the frequency of certain API call types within the API call category between multiple samples. This helps the analyst compare different variants of ransomware and determine whether they act differently in their malicious operations. This visualization captures the categorical aspects of a sample.

FIGURE 4. RanViz treemap.

2) TIME SERIES

The time series visualization allows the user to see the occurrence of different categories of API calls over time in the sample as depicted in Figure 5. The time series shows the count of an API call category at a specific point in time. Time series visualization also allows the user to filter and toggle off the categories that they do not want to display and focus on specific categories. The user is able to zoom in and out of the Time Series to focus closely on densely displayed activity. The axes adjust when the user zooms in allowing them to see activity in smaller time intervals. The user can also select up to five API call types for visualization in another time series to see the chronological order and temporal insights of the specific API call types. These interactions and adaptability ensure that the analyst can easily understand the information presented through the visualization. Individual time series analysis of a sample allows the analyst to reconstruct the sequence of events of malicious actions that the sample executes and the time taken for different activities to unfold for digital forensic purposes. For example, the analyst will be able to see that a sample first generates the encryption key and then encrypts the files after 5 seconds. For sample comparison, the user can see the differences in activity spikes of API call categories in multiple samples and compare differences among variants. This visualization captures both the categorical and temporal aspects of a sample.

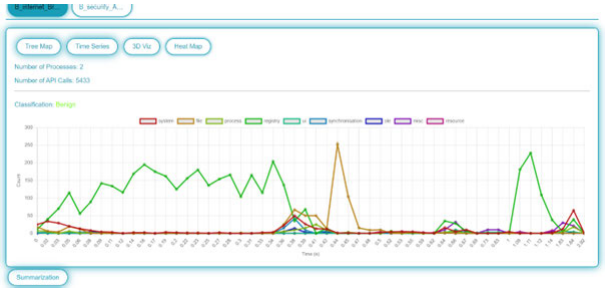


FIGURE 5. RanViz time series.

3) 3D VISUALIZATION

The 3D visualization allows the user to see the occurrence of different categories of API calls over time with the added dimension to visualize all the categories occurring simultaneously with a high-oblique view of the activity, as depicted in Figure 6. The count of an API call category at a specific timestamp is shown by the height of the 3D bar, allowing the user to visualize spikes in the activity. The individual analysis of a sample allows the user to see which categories of API calls have spikes in activity and at which point in time in the sample execution. For example,

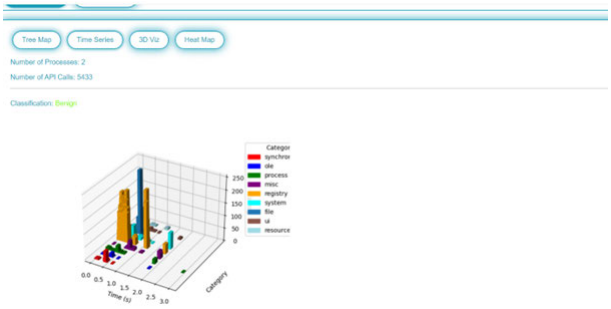


FIGURE 6. RanViz 3D visualization.

analysts will be able to see that registry-related activities were prevalent throughout the sample execution period. For sample comparison, the user can see the differences in the activity of API call categories in multiple samples and compare differences among variants. This visualization captures both the categorical and temporal aspects of a sample.

4) HEATMAP

The heatmap visualizes the intensity of the occurrences of different categories of API calls categories over time, as depicted in Figure 7. There are four categories of intensity of occurrence for API call categories highlighted in different colours. For individual analysis, the user can see the intensity of the occurrence of categories of API calls and at what point in time the intensity of the categories occurred in the sample execution. For example, the analysts will be able to see that low-intensity activity was prevalent in a sample, and high-intensity file-related activity occurred in the middle of the execution period, indicating a lot of encryption midway in the execution. For sample comparison, the user can compare the intensity of occurrence of API categories between multiple samples and uncover the category that can be an indicator of malicious behaviour. This visualization captures both the categorical and temporal aspects of a sample.

FIGURE 7. RanViz heatmap.

5) SUMMARIZATION

Summarization can be viewed by the user by selecting the summarization button. Summarization utilizes the MalFe dataset to summarize the behaviour of the ransomware and benign samples. The average frequency for each API call type occurring in the ransomware and benign samples are

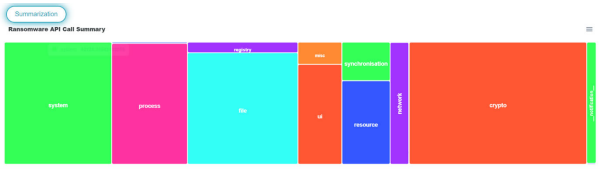


FIGURE 8. RanViz ransomware summarization.

FIGURE 9. RanViz benign summarization.

visualized in a treemap, as shown in Figures 8 and 9. This helps the user uncover the overall differences between the ransomware and benign samples. The user can see which API call types occur with the highest frequency in ransomware and look out for these when they analyze their uploaded samples. The different API call types are represented with different colours to assist the user in differentiating between them. The use of summarization allows the system to leverage the dataset used for training the machine learning models for visualization. The next section describes the machine learning feature extraction and model training.

E. RanViz USABILITY EVALUATION

RanViz was evaluated using Jakob Nielsen's 10 Usability Heuristics [45] as seen in Table 1. These are 10 principles that can be used to evaluate the user interface and usability of a system. RanViz performed well in the evaluation, meeting 9 out of 10 of the usability heuristics. Systems that have a score of 8 and above are considered to be excellent [45].

VI. RanViz MACHINE LEARNING

A. FEATURE EXTRACTION

MalFe reports were used as the dataset for training the machine learning models. Features were extracted from the behaviour section of the reports and across multiple processes for each report. The greatest challenge in preprocessing samples was that each MalFe report had a widely varying number of API calls, with some reports having an insurmountable number of API calls. To resolve this, a feature extraction approach was used to simplify the samples while maintaining all important data in the sample. The API call entries in each report had a standard set of attributes. The attributes of the API calls are listed in Table 2.

For the purposes of this research the Category, Api, and time were the relevant attributes. The MalFe JSON reports were preprocessed, and irrelevant data attributes were removed, leaving API calls for each report. All the unique types of API calls in the entire MalFe dataset of 3082 reports of both ransomware and benign samples were extracted and used to create a Python dictionary that acts as a template

TABLE 1. RanViz Usability Heuristic Evaluation.

Usability Heuristic	Does the system meet this usability heuristic?	RanViz feature that meets criteria
Visibility of system status	Yes	RanViz keeps user informed about what is going on and gives feedback in a timely manner.
Match between system and the real world	Yes	RanViz follows real world conventions in the words the system uses and its metrics
User control and freedom	Yes	Users are able to upload the number of samples of their choice and select the visualizations that they way.
Consistency and standards	Yes	There is consistency in the colours used to represent certain API call categories and API call types.
Error prevention	Yes	RanViz prevents the upload of samples that are not of the correct format
Recognition rather than recall	Yes	The options to upload samples and select visualizations are clearly visible in a single web page even after selecting a specific visualization meaning the user does not have to remember options.
Flexibility and efficiency of use	Yes	RanViz allows users to filter categories in the Time Series and Tree Map visualizations and see information that they want to see.
Aesthetic and minimalist design	Yes	The design does not have unnecessary information and has a simple design with most of the colours being used in the visualizations themselves and not the system itself.
Help users recognize, diagnose, and recover from errors	Yes	RanViz provides feedback to the user who uploads invalid samples and shows a message specifying what the problem with the sample is. For example: JSON error, No API calls .
Help and documentation	No	There is no help or documentation provided by the system

TABLE 2. MalFe report API call data attributes.

Category	The category that the API call type belongs to
Status	This shows the status of the API call as a success or failure
Stacktrace	In the event of an error this contains information during the execution of the API call that can be used to debug
Api	The API call type that has been executed
return_value	This is the value returned after the API call
arguments	These are the arguments that are passed to the API call
time	The Unix timestamp of the API call execution
tid	The thread id of the thread that executed the API call
flags	These are additional flags or options that are related to the API call

for all sample features to enable training in machine-learning models. This was done because of the varying number of API calls in each sample, which would have required padding or truncating to be trained in a machine-learning model. In total, 280 unique API call types were identified and used in the Python dictionary template. For feature extraction, one-hot encoding was explored, but it created a problem where the samples needed to be padded or truncated because of the varying array lengths. This would result in the loss or modification of important data.

Frequency encoding was also explored and it worked with the Python API call dictionary template without any loss or modification of data. For each of the unique 280 API calls, the frequency of their occurrence in the sample was tallied. This allowed the machine-learning models to be trained based on the API call frequency. This was represented in a CSV file containing entries for every MalFe report both ransomware and benign using the dataset generation[1] feature from MalFe. To train the models based on the temporal aspects of the behaviour of samples, the total time interval between the occurrences of the unique 280 Api types in each sample was calculated. This was also represented in a CSV file containing entries for every MalFe report, both ransomware and benign, using the dataset generation[2] feature from MalFe. This allowed the machine learning models to be trained based on the API call temporal interval.

Binary encoded strings were generated for the 280 unique API call types in each sample to train the models based on the sequential occurrence of API calls. When the API call type occurred in an entry in the sample, it was represented by a '1', and if it did not occur in an entry, it was represented by '0'. This was then concatenated into a sequence string for each API call type within each sample. Label Encoding did not work for the sequence strings because of the large variation in the sequence string combinations, resulting in fewer common strings. Run-length encoding was explored; however, it failed due to the extremely large and varying values it created, and this did not simplify the complexity of the sequence strings. NGRAMs were used to further process the sequence strings to represent them with simpler and common sequence strings by uncovering common sequence patterns in the sequence strings. This allowed Label Encoding to be used correctly, with few variations in the sequence string combinations. For this, a sample size of 2770 was used because of problems with processing the sequence strings for extremely large MalFe reports, which led to them being excluded. This was also represented in a CSV file containing entries for MalFe reports, both ransomware and benign, using the dataset generation feature from MalFe, but this was done on a local machine due to memory issues on the MalFe servers. This allowed the machine-learning models to be trained based on the API call sequence.

In the end, the feature extraction focused on three categories: API call frequency, API call temporal interval, and API call sequence. This covers both the categorical and temporal behaviour of API calls with respect to the machine learning section of this research. The CSV files for the frequency and temporal features had 285 columns and 3083 rows. The CSV file for the sequential feature had 285 columns and 2771 rows.

B. MACHINE LEARNING MODELS TRAINING

After the features were extracted and represented in CSV files, they were used to train the machine learning models. The sample name, sha256, and category columns were

[1]https://malfe.cs.up.ac.za/datasets/view/API CALL FREQUENCY/42
[2]https://malfe.cs.up.ac.za/datasets/view/API Temporal Interval/44

dropped, leaving the label column as the target column and the API call columns as the dependent variable columns. The dataset was split into 60% training and 40% testing sets. The supervised learning models that were explored were the Long Short Term Memory model (LSTM), K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Logistic Regression (LR), and Convolutional Neural Network (CNN). The LSTM was explored because of its ability to work with time series problems. The KNN was explored because of its ability to classify unknown data into categories based on common features that are appropriate for the classification of ransomware and benign samples. The SVM was explored because of its ability to work with high-dimensional data. Logistic Regression was explored because it works well with linearly separable data and is well-suited for binary classification, which is appropriate for the classification of samples as either ransomware or benign. The CNN was explored because it can uncover structures and features within the dataset without the requirement of feature engineering. The parameters used in each model are listed in Table 3. The next section is the evaluation.

TABLE 3. Model parameters.

Model	Parameters
LR	solver='lbfgs', 'liblinear', max_iter=10000
KNN	n_neighbors=2
SVM	kernel='linear'
CNN	optimizer='adam', loss='binary_crossentropy'
LSTM	optimizer='adam', loss='binary_crossentropy'

VII. EVALUATION OF RanViz SYSTEM AND MACHINE LEARNING MODELS

A. RanViz VALIDATION

The NIST validation cycle [46] was used to validate RanViz. The cycle involves defining the requirements, test assertions for how the system should behave, test cases for testing the requirements, and a test methodology. At the end of the cycle, validation testing was conducted to validate the system [42]. The requirements are divided into Core Requirements (CR), which are listed in Table 4 and Optional Requirements (OR), which are listed in Table 5. Labels for each of the requirements are then used in the compliance matrix, which is at the end of the validation cycle of the system.

TABLE 4. Core requirements.

Label	Description
CR-01	Users should be able to upload MalFe Reports
CR-02	Users should be able to see the classification of the uploaded samples
CR-03	Users should be able to select the visualization of the sample
CR-04	Users must see highlighted malicious API calls
CR-05	Users must be able to filter categories in visualizations
CR-06	Users must be able to compare multiple samples
CR-07	Users must be able to see summarization of ransomware and benign dataset

TABLE 5. Optional requirements.

Label	Description
OR-01	Users should be able to download the selected visualizations
OR-02	Users should be able to see basic information about processes and API calls in each sample

The test assertions are the postconditions for the various functions of the system. The test assertions are listed in Table 6.

TABLE 6. Test assertions.

Label	Description
TA-01	The system shall detect upload errors and provide error messages. **Justification:** To prevent invalid uploads and provide the user with feedback.
TA-02	The classification of the sample must be computed and assigned a correct label **Justification:** To provide valid output of the integrated machine learning model and classify uploaded samples
TA-03	The corresponding file type must be generated by the system **Justification:** To allow the user to download the correct file type that they require
TA-04	The filtered-out categories should be removed from the view of the user **Justification:** To allow the user to focus on the categories that they want to view

The test cases are defined and listed in Table 7. These are for the purpose of validating that the system meets the standards and test assertions.

TABLE 7. Test cases.

Label	Description
TC-01	Upload Malfe reports
TC-02	Switch between tabs of multiple samples for comparison
TC-03	View the classification of each of the uploaded samples
TC-04	View the treemap visualization
TC-05	Select a category in the treemap visualization
TC-06	View the Time series visualization
TC-07	Filter categories in the time series visualization
TC-08	View the 3D visualization
TC-09	View the Heatmap visualization
TC-10	Download a visualization
TC-11	View summarization visualization
TC-12	Select summarization category

The compliance matrix is the mapping of the system requirements to the test case(s) that meet the test assertions. In certain cases, the test assertions are included in the result column when they are met, but in other cases where a manual check occurred, a -check- indication is used to show that the result is compliant. Manual checks and test assertions can also occur simultaneously. The compliance matrix presented in Table 8 confirms that the test assertions were fulfilled, therefore validating the system.

TABLE 8. Compliance matrix.

No	Requirement	Test Case	Result
1	CR-01	TC-01	TA-01, -check-
2	CR-02	TC-02, TC-03	TA-03, -check-
3	CR-03, CR-04	TC-04, TC-06, TC-08, TC-09	-check-
4	CR-05	TC-05, TC-07	TA-04, -check-
5	CR-06	TC-02	-check-
6	CR-07	TC-11, TC-12	-check-
7	OR-01	TC-10	TA-03
8	OR-02	TC-03	-check-

B. MACHINE LEARNING MODELS EVALUATION

The results of the machine learning section of this research are presented in tables for each of the three feature categories. The models were evaluated based on the following metrics: accuracy, confidence score, precision, F1-score, latency, recall, and AUC ROC. Accuracy is a measure of how frequently the model makes correct predictions. It is expressed as the tally of correct predictions the model makes divided by the tally of all predictions made by the model. The confidence score is a probability score that indicates how confident the model is in making the correct prediction. Precision refers to the measure of how frequently the model makes correct predictions of the positive class. It is expressed as the tally of true positives divided by the sum of false positives and true positives. Recall is a measure of how frequently the model correctly identifies true positives from all the actual positives in the dataset. It is expressed as the tally of true positives divided by the sum of false negatives and true positives. Latency refers to the time taken by the deployed model to make a prediction on an unseen sample. The The F1-score is a measure of the harmonic mean of the recall and precision, making it an important metric because it represents both recall and precision. The AUC ROC is a measure of the ability of the model to differentiate between positive and negative classes. It is also an important metric because it provides a sense of how good the model is beyond just assessing the accuracy. The results of the machine learning section of this research are presented in tables for each of the three feature categories.

1) API CALL FREQUENCY

For API call frequency all the models (LR, KNN, SVM, CNN, LSTM) presented in Table 9 achieved accuracy in the range 96 – 98%. The confidence scores for the LR and KNN models were higher than those for the SVM, CNN, and LSTM models. This indicates that the LR and KNN models were more confident that they made the correct predictions. All the models achieved precision scores within the range 0.98-1.0 with CNN and LSTM achieving scores of 1.0. These high precision scores indicate that the models have a low false positive rate and make consistent predictions. However, both the CNN and LSTM models had recall scores of 0.0, indicating that these models were actually failing to identify any of the positives. This is possible because of high conservativeness or an imbalanced dataset, which is the case for the MalFe dataset. For LSTM, this could be due to relying on sequence length, which is not available in this case, because all the data followed a fixed 280 API calls template. LR, KNN, and SVM achieved high recall scores within the range 0.98-0.99 indicating that they were correctly identifying the actual positive classes with fewer false negatives and are effective at making the correct detections. LR, SVM, and KNN achieved a high F1-score, which indicates that the models have a balanced performance in terms of precision and recall. Consequently, CNN and LSTM achieved poor F1-scores of 0.0 due to their poor Recall scores. In terms of AUC ROC, all models achieved high scores as depicted in Figure 10. This indicates that the models are able to distinguish between the positive and negative classes. For LR, KNN, and SVM the models performed very well in the key metrics of the F1-score and AUC ROC. In terms of latency, the model that outperformed the other good models(KNN and SVM) was Logistic Regression with a latency of 0.34 ms. It also has a higher confidence score than the SVM, which is the closest-performing model in terms of latency. Latency is important in the integration of the model with RanViz, because of time and efficiency are important in the analysis of a large amount of data in digital forensics. This leads to the conclusion that Logistic Regression is the best model in terms of API call frequency.

TABLE 9. Results of the machine learning models for API call frequency.

	Accuracy	Confidence score	F1 score	Precision	Latency (ms)	Recall	AUC ROC
LR	0.96	0.98	0.98	0.98	0.34	0.99	0.96
KNN	0.97	0.99	0.98	0.98	1.44	0.99	0.95
SVM	0.97	0.88	0.98	0.98	0.35	0.98	0.96
CNN	0.98	0.82	0.0	1.0	81.40	0.0	0.96
LSTM	0.98	0.83	0.0	1.0	82.65	0.0	0.99

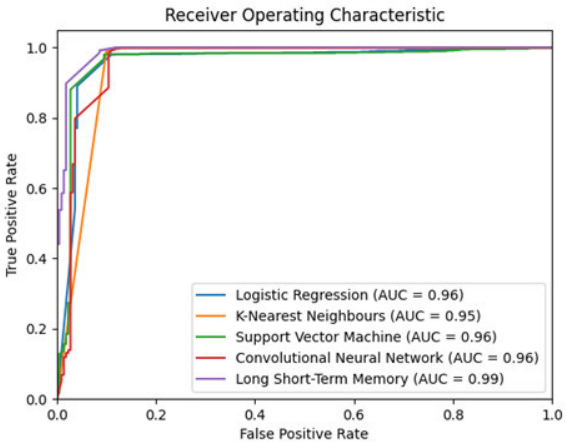


FIGURE 10. API Call frequency AUC ROC.

2) API CALL TEMPORAL INTERVAL

For API call temporal interval all the models (LR, KNN, SVM, CNN, LSTM) presented in Table 10 achieved accuracy within the range 95-97%. The confidence scores for the

TABLE 10. Results of the machine learning for API call temporal interval.

	Accuracy	Confidence score	F1 score	Precision	Latency (ms)	Recall	AUC ROC
LR	0.97	0.98	0.98	0.98	0.84	0.98	0.97
KNN	0.97	0.99	0.98	0.98	2.86	0.98	0.95
SVM	0.97	0.72	0.98	0.98	0.89	0.98	0.97
CNN	0.96	0.84	0.0	1.0	72.17	0.0	0.95
LSTM	0.95	0.80	0.0	1.0	77.82	0.0	0.96

LR and KNN models were higher than those for the SVM, CNN, and LSTM models. This indicates that the LR and KNN models were more confident that they made the correct predictions. All the models achieved precision scores within the range 0.98-1.0 with CNN and LSTM achieving scores of 1.0. These high precision scores indicate that the models have a low false positive rate and make consistent predictions. However, both the CNN and LSTM models both had recall scores of 0.0 indicating that these models were actually failing to identify any of the positives. This is possible because of high conservativeness or an imbalanced dataset, which is the case for the MalFe dataset. For LSTM, this could be due to relying on sequence length which is not available in this case, because all the data followed a fixed 280 API calls template. LR, KNN, and SVM achieved high recall scores indicating that they are correctly identifying the actual positive classes with fewer false negatives and are effective at making the correct detections. LR, SVM, and KNN achieved a high F1-score which indicates that the models have a balanced performance in terms of precision and recall. Consequently, CNN and LSTM achieved poor F1 scores of 0.0 due to their poor recall scores. In terms of AUC ROC, all models achieved high scores as depicted in Figure 11, with LR and KNN achieving the joint highest at 0.97. This indicates that the models are able to distinguish between the positive and negative classes. For LR, KNN, and SVM the models performed very well in the key metrics of the F1-score and AUC ROC. In terms of latency, the model that outperformed the other good models(KNN and SVM) was Logistic Regression with a latency of 0.84 ms. It also has a higher confidence score than the SVM, which is the closest-performing model in terms of latency. This leads to the conclusion that Logistic Regression is also the best model in terms of the API call temporal interval. The results for the API call temporal intervals were similar to those for the API call frequency.

3) API CALL SEQUENCE

For API call sequence all the models (LR, KNN, SVM, CNN, LSTM) presented in Table 11 achieved accuracy within the range 98-99%. The confidence scores for the LR, SVM, and KNN models were higher than those for the CNN and LSTM models. This indicates that the LR and KNN models were more confident that they made the correct predictions. All models achieved precision scores within the range of 0.99-1.0 with CNN and LSTM achieved scores of 1.0. These high precision scores indicate that the models have

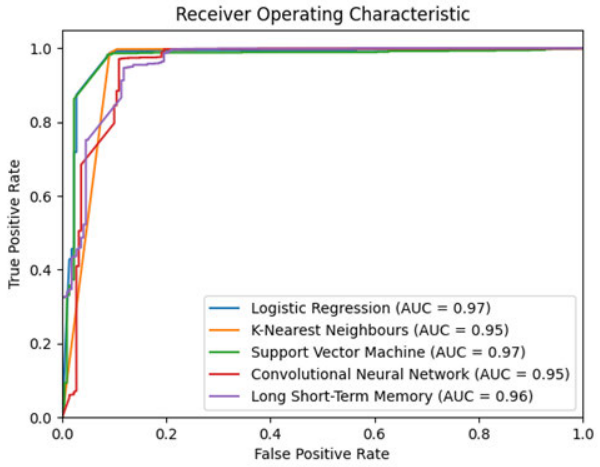


FIGURE 11. API Call temporal interval AUC ROC.

TABLE 11. Results of the machine learning models for API call sequence.

	Accuracy	Confidence score	F1 score	Precision	Latency (ms)	Recall	AUC ROC
LR	0.98	0.99	0.99	0.99	0.96	0.99	0.96
KNN	0.98	0.99	0.99	0.99	10.67	0.98	0.97
SVM	0.99	0.99	0.99	0.99	0.15	0.99	0.98
CNN	0.99	0.85	0.0	1.0	82.03	0.0	0.98
LSTM	0.99	0.85	0.0	1.0	87.23	0.0	0.99

a low false positive rate and make consistent predictions. However, both the CNN and LSTM models had recall scores of 0.0, indicating that these models were actually failing to identify any of the positives. This is possibly because of high conservativeness or an imbalanced dataset, which is the case for the MalFe dataset. For LSTM, this could be due to relying on sequence length, which is not available in this case, because all the data followed a fixed 280 API calls template. LR, KNN, and SVM achieved high recall scores, indicating that they were correctly identifying the actual positive classes with fewer false negatives and are effective at making the correct detections. LR, SVM, and KNN achieved a high F1-score, which indicates that the models have a balanced performance in terms of the precision and recall. Consequently, CNN and LSTM achieved poor F1- scores of 0.0 due to their poor recall scores. In terms of AUC ROC, all models achieved high scores, as depicted in Figure 12, with LSTM achieving the highest at 0.99. SVM outperformed LR and KNN in terms of AUC ROC with a score of 0.98. In terms of latency, the model that outperformed the other good models (KNN and LR) was SVM, with a latency of 0.15 ms. In this case, SVM outperforms LR in terms of latency, which is different from the previous two cases. This leads to the conclusion that the Support Vector Machine is the best model in terms of the API call sequence. The next section presents case studies for the evaluation of RanViz.

RanViz models consistently outperformed other existing models and techniques for ransomware detection in the

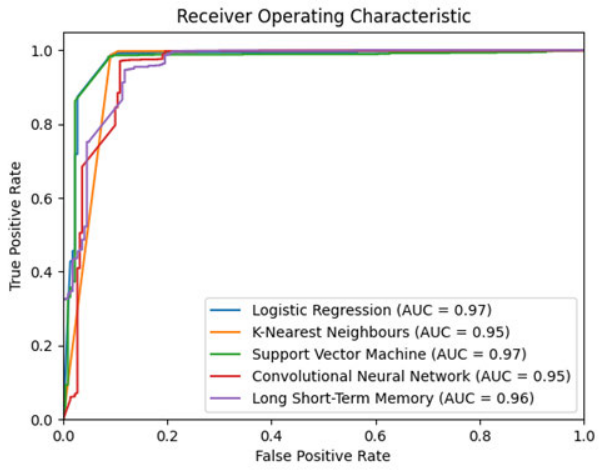


FIGURE 12. API call sequence AUC ROC.

TABLE 12. Comparison of results.

Reference	Model	Accuracy	Latency (ms)	Recall	Precision	F1-Score
Frequency	LR	0.96	0.34	0.99	0.98	0.98
Temporal Interval	LR	0.97	0.84	0.98	0.98	0.98
Sequence	SVM	0.99	0.15	0.99	0.99	0.99
[38]	Cooperative clustering	0.97	120	-	-	-
[20]	Aggregated Random Forest	0.94	0.15	0.95	0.94	0.94
[28]	LSTM	0.99	-	0.98	0.99	0.99
[14]	RLR	0.96	-	-	-	-

metrics of Accuracy, Recall, precision and F1-Score as depicted in Table 12. The Aggregated Random Forest has better latency than two of the models used in RanViz because it was developed to work in a distributed network where response time will be crucial. The cooperative clustering performed poorly because it uses multiple detection agents and has a feature extraction module as part of the model architecture. The RanViz SVM model does have the same latency as the Aggregated Random Forest and outperforms it in the metrics of Accuracy, Recall, Precision and F1-Score. The RanViz SVM matches the performance of the LSTM by Sgandurra [14] in the metrics of Accuracy, Recall and F1-Score. The RanViz SVM outperforms the LSTM in terms of precision.

C. CASE STUDIES

Case studies were conducted to evaluate RanViz as a ransomware analysis platform and compare it with existing malware analysis systems: MalView, AnyRun [47], and Hybrid [48]. To the best of the authors' knowledge, there are no publicly available prototypes for KAMAS and Eventpad. The aim of these case studies is to show that RanViz is able to correctly visualize and identify behaviour associated with certain variants of ransomware and will be valuable for representing and identifying the behaviour of future unknown variants of ransomware.

1) GrandCrab

GrandCrab is a crypto-ransomware that is spread as Ransomware as a Service (RaaS). GrandCrab uses a Salsa stream cipher with the RSA Algorithm. Before GrandCrab encrypts the files of the victim, it must first search for and terminate the background processes that attempt to obtain the server IP address [49]. The payment of the ransom is done using cryptocurrencies such as Bitcoin.

A sample of a GrandCrab report was obtained from the MalFe dataset. The sample was then converted into a procmon output log csv to match the input required by MalView. Each process of the sample were extracted. This includes the process name, process path, process id, first_seen time, and operations that represent function calls that occur in each process summary were represented as part of the operations entry in the procmon log. The procmon log csv for GrandCrab was used as input to MalView. The output of the GrandCrab sample in MalView is shown in Figure 13. The system shows that GrandCrab has operations in the FileSystem and Registry categories. The system fails to group and highlight other related operations choosing to focus on the commonly encountered operations, which is a disadvantage when a sample is dominated by operations that are not commonly encountered. The system does not show behaviour related to the termination of processes before encryption, which is a characteristic of GrandCrab. The system visualizes file related activity which indicates the encryption behaviour of the sample. The system works better in cases where the samples have many processes with a lot of operations but fails to visualize a lot of activity in its time series because this GrandCrab sample has only nine processes.

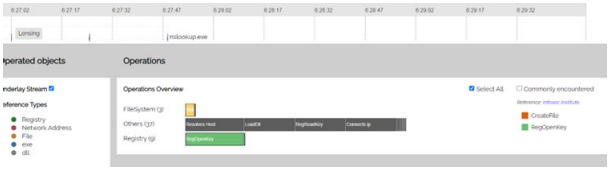


FIGURE 13. MalView GrandCrab output.

A GrandCrab ransomware executable was obtained from the Intelligent Cyber Forensics Lab (ICFL) repository and used as input for Hybrid and AnyRun. The generated reports from Hybrid[3] and AnyRun[4] do not have visualizations. The same sample of a GrandCrab report was obtained from the MalFe dataset. The sample was then used as input to RanViz. The output of the GrandCrab sample in RanViz is shown in Figure 14. The system shows that GrandCrab has nine processes and in addition, it shows that it has 7741 API calls. The correct classification output of the sample as ransomware from the integrated machine-learning models is also shown in red.

[3]https://tinyurl.com/23e3xc57
[4]https://app.any.run/tasks/faa3dcbd-4f9b-43bc-90ad-153371778979

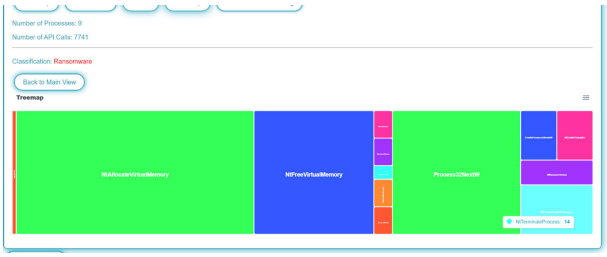


FIGURE 14. RanViz GrandCrab treemap process drilldown.

Selecting the process category in the treemap shows the drill-down of process-related API call types executed by GrandCrab. NtTerminateProcess appears in the visualization which indicates that GrandCrab terminated processes 14 times, as depicted in Figure 14. Terminating background processes is one of the characteristics of GrandCrab that RanViz can visualize.

FIGURE 15. RanViz GrandCrab treemap crypto drilldown.

Selecting the crypto category in the treemap shows the drill-down of cryptographic-related API call types executed by GrandCrab. CryptGenKey appears in the visualization, which indicates that GrandCrab made an API Call to generate encryption, as depicted in Figure 15. This is one of the characteristics of GrandCrab to utilize the RSA encryption algorithm that RanViz visualizes for the analyst that MalView does not do because it only shows file related activity but does not show that a cryptographic key was generated.

2) SODINOKIBI

Sodinokibi is a crypto-ransomware that uses a combined attack model with Ransomware as a Service (RaaS). It is characterized by the encryption of files and the deletion of backup files. It can spread to other computers and does not need to connect to the command-and-control server to carry out malicious activity [50].

A sample of a Sodinokibi report was obtained from the MalFe dataset. The sample was then converted into a procmon output log csv to match the input required by MalView. The procmon log csv for Sodinokibi was used as input to MalView. The output of the Sodinokibi sample in MalView is shown in Figure 16. The system shows that Sodinokibi has operations in the Registry and FileSystem categories that were executed. The system also shows behaviour related to the deletion and encryption of files

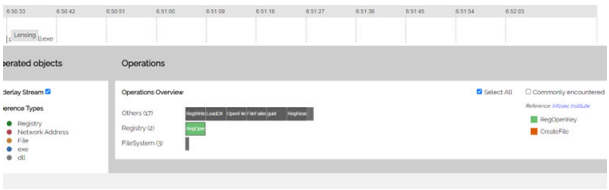


FIGURE 16. MalView sodinokibi output.

among the FileSystem category but does not show anything related to the behaviour of spreading the ransomware. The system also fails to group and highlight other related operations choosing to focus on the commonly encountered operations, which is a disadvantage when a sample is dominated by operations that are not commonly encountered. The system works better in cases where the samples have many processes with a lot of operations but fails to visualize a lot of activity in its time series due to this Sodinokibi sample only having three processes.

A Sodinokibi ransomware executable was obtained from the ICFL repository and used as input to Hybrid and AnyRun. The generated reports from Hybrid[5] and AnyRun[6] do not have visualizations. The same sample of a Sodinokibi report was obtained from the MalFe dataset. The sample was then used as input to RanViz. The output of the Sodinokibi sample in RanViz is shown in Figure 17. The system also shows that Sodinokibi has three processes and in addition, it shows that it has 13237 API calls. The correct classification output of the sample as ransomware from the integrated machine-learning models is also shown in red.

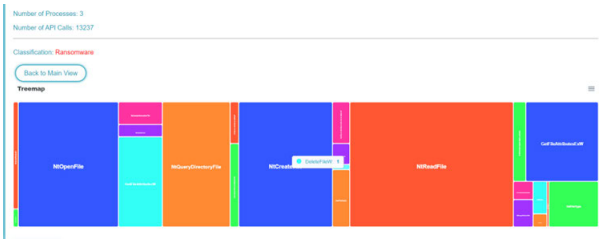


FIGURE 17. RanViz Sodinokibi treemap file drilldown.

Selecting the file category in the treemap shows the drill-down of file-related API call types that were executed by Sodinokibi. DeleteFileW appears in the visualization which indicates that Sodinokibi made an API call to delete a file, as depicted in Figure 17. File deletion is one of the characteristics of Sodinokibi which RanViz can visualize, while AnyRun and Hybrid do not. Selecting the Crypto category in the treemap shows the drill-down of encryption-related API call types that were executed by Sodinokibi. CryptExportKey, CryptHashData, and CryptCreateHash appear in the visualization which indicates that Sodinokibi made an API calls to encrypt files as depicted in Figure 18.

[5]https://tinyurl.com/2cj9acug
[6]https://app.any.run/tasks/5d9b210f-2783-4e35-a824-41c28ad47abd

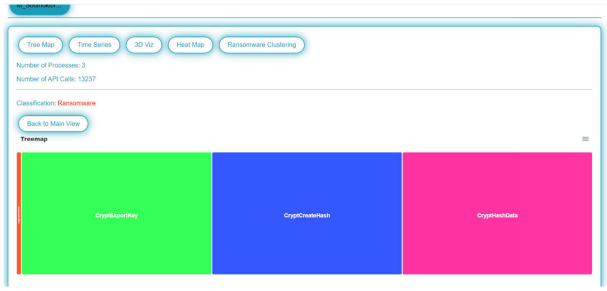


FIGURE 18. RanViz sodinokibi treemap crypto drilldown.

File encryption is also a characteristic of Sodinokibi that RanViz can visualize.

Selecting the time series visualization and filtering to the process category shows that there were many process related activities towards the end of the operations performed by Sodinokibi including NtOpenProcess and allocation of virtual memory, indicating that many processes were active to spread the ransomware after it completed its encryption, as depicted in Figure 19. There is also an absence of network-related activity, indicating that Sodinokibi does not connect with the command-and-control server. RanViz can visualize these characteristics whilst MalView, Hybrid, and AnyRun do not.

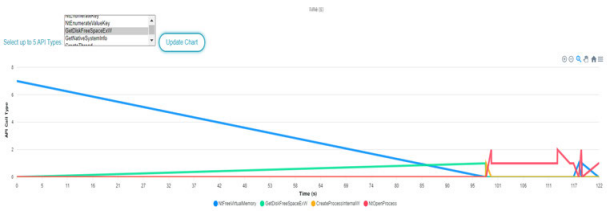


FIGURE 19. RanViz sodinokibi process time series.

3) PETYA

Petya is a ransomware that targets and modifies the Windows Master Boot Record thereby causing the system to crash. After the victim reboots the system, the modified Master Boot Record prevents the system from loading and instead displays the ransom message [51]. Although this exhibits locker ransomware characteristics there are also variants of Petya that are classified as crypto ransomware.

A sample of a Petya report was obtained from the MalFe dataset. The sample was then converted into a procmon output log csv to match the input required by MalView. The procmon log csv for Petya was used as input to MalView. The output of the Petya sample in MalView is shown in Figure 20. The system shows that Petya has operations in the FileSystem category. There is sparse activity shown for this sample due to Petya only having two processes. It does not show any behaviour related to the modification of the Master Boot Record which is the main characteristic of Petya.

A Petya ransomware executable was obtained from the ICFL repository and used as input to Hybrid and AnyRun.

FIGURE 20. MalView petya output.

The generated reports from Hybrid[7] and AnyRun[8] do not have visualizations. The same sample of a Petya report was obtained from the MalFe dataset. The sample was then used as input to RanViz. The output of the Petya sample on RanViz is shown in Figure 21. The system shows that Petya has 2 processes and in addition it shows that it has 353 API calls. The correct classification output of the sample as ransomware from the integrated machine learning models is also shown in red. By selecting the treemap visualization and seleting the file category the user can see that the sample made DeviceIoControl API calls which indicate direct drive access, multiple SetFilePointerEx API calls that are used to set the file pointer to the beginning of the Master Boot Record, and in conjunction with multiple NtWriteFile and NtCreateFile API calls, which indicate modification of the Master Boot Record after accessing it.

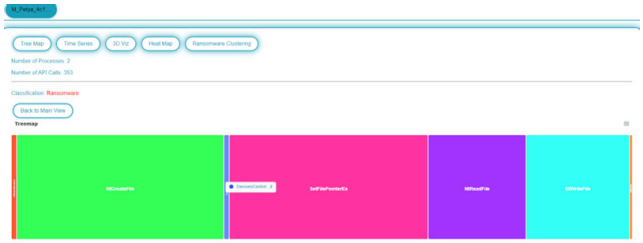


FIGURE 21. RanViz petya treemap file drilldown.

By selecting the time series visualization, the user can confirm this behaviour by observing the sequence of when these API call types occurred and the time taken between the occurrences. DeviceIoControl occurs in the beginning, followed by a delay with low activity, and SetFilePointerEx occurs before NtWriteFile. NtWriteFile and NtCreatFile have a spike in activity around the same time as when SetFile-PointerEx has a spike in activity, as depicted in Figure 22. RanViz is able to correctly visualize the Master Boot Record modification behaviour of Petya while MalView, Hybrid, and AnyRun do not.

None of the systems compared to RanViz in the case studies were able to correctly visualize and identify the entire behaviour associated with all the variants of ransomware used in the case studies whereas RanViz was able to. The next section provides a detailed comparative analysis of the features offered by RanViz and the other visualization systems.

[7]https://tinyurl.com/2bdv8npt
[8]https://app.any.run/tasks/6480487a-e09f-435d-852a-5238dac727f4

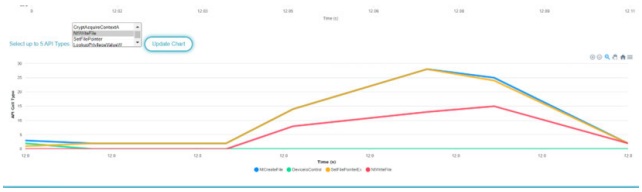


FIGURE 22. RanViz petya time series.

D. COMPARATIVE ANALYSIS

RanViz offers features similar to those of other systems and additional unique features that enhance the depth of ransomware analysis, as shown in Table 13. The unique features of RanViz include the ability to use integrated machine learning models to classify the uploaded sample as benign or ransomware based on API call-related features. MalView does not classify the uploaded sample as malicious or benign but instead classifies the domains to which the sample is connected. EventPad and KAMAS do not have classification features at all. This feature is crucial for classifying future unknown variants of ransomware and helps analysts classify variants that mask their activity. Another key unique feature of RanViz is the API call categorical and temporal analysis. RanViz has visualizations that show analysts the behaviour of different categories and types of API calls. It also includes temporal aspects to show analysts the sequence of events and the time taken for different activities to unfold. MalView focuses on process analysis, which does not provide the depth required to uncover ransomware behaviour, as demonstrated in the case studies. Eventpad focuses on network activity which is useful for understanding network-related activities but does not go to the depths of the internal operations of the ransomware. KAMAS does have API call-related visualizations, but it does not provide categorical and temporal visualizations to analysts and focuses on the occurrences of API calls. AnyRun and Hybrid do not have any visualizations. RanViz also has the summarization of the dataset used by the system to train the integrated machine learning models.

MalView, KAMAS, Eventpad, AnyRun, and Hybrid do not use any dataset and do not provide a summarization of a dataset. Summarization is important to help analysts gain insight into the common behaviour that ransomware exhibits based on the dataset. RanViz allows analysts to upload multiple samples simultaneously and compare their visualizations, thereby enhancing the comparison capabilities. MalView, KAMAS, Eventpad, AnyRun, and Hybrid do not allow simultaneous upload and comparison of multiple samples.

VIII. REAL WORLD APPLICATION

RanViz can be effectively applied in an enterprise network as part of the incident response process, particularly in post-incident investigations. It assists the response team in analyzing ransomware incidents by visualizing key attack behaviors. While RanViz has the capability to process data in real-time, generating meaningful visualizations requires collecting and analyzing data over a specific timeframe. This makes it more suited for forensic analysis rather than immediate ransomware detection.

In an enterprise setting, integrating RanViz with existing cybersecurity solutions, such as Security Information and Event Management (SIEM) and Endpoint Detection and Response (EDR), enhances its effectiveness. SIEM systems detect ransomware activity, while EDR helps pinpoint the malicious executable responsible for the attack. During the forensic phase, the ransomware sample can be analyzed using MalFe, which generates a detailed report. This report is then processed by RanViz, where machine learning models classify the sample as ransomware and visualize its behaviour.

Through RanViz, organizations can gain insights into how the ransomware spreads—whether it propagates through connected devices, exploits specific operating systems, or communicates with a command-and-control server. Additionally, correlating RanViz's visualizations with SIEM logs provides a clearer picture of the attack, helping organizations strengthen their defences and mitigate future risks.

TABLE 13. Comparative analysis.

Features	Kamas	Eventpad	MalView	AnyRun	Hybrid	RanViz
Possesses filtering capabilities?	Yes	Yes	Yes	No	No	Yes
Provides brief information about uploaded sample?	Yes	Yes	Yes	Yes	Yes	Yes
Highlights malicious indicators?	Yes	Yes	Yes	Yes	Yes	Yes
Has behavioural visualizations?	Yes	Yes	Yes	No	No	Yes
Shows categorized activity?	No	No	Yes	Yes	Yes	Yes
Has a variety of interactive visualizations and analysis?	No	Yes	Yes	No	No	Yes
Has manual encoding of samples?	No	Yes	No	No	No	No
Possesses multiple sample upload and comparison capabilities?	No	No	No	No	No	Yes
Integration of machine learning for classification?	No	No	No	No	No	Yes
API call categorical and temporal analysis?	No	No	No	No	No	Yes
Provides summarization of dataset?	No	No	No	No	No	Yes

IX. CONCLUSION

RanViz offers effective ransomware analysis by providing visualizations based on the API call time-series categorical behaviour. It also applies machine learning to provide classification capabilities based on API call behaviour. This unique approach will help in the understanding of ransomware operations and behavioural traits. It also shows that using time-series and categorical API call behaviour has value in the understanding of ransomware variant behaviour. This will help analysts handle the complexity of ransomware and the large volume of API calls that they present. The machine learning models explored in the development of RanViz performed well in classifying samples based on the API call features with each model having accuracy and AUC

ROC scores greater than 95%, which provides confidence in the accuracy of the integrated machine learning models used by RanViz. To evaluate its effectiveness, RanViz was compared with other existing malware visualization systems through case studies and a comparative analysis. It was successfully validated using the NIST validation cycle and correctly visualized the characteristics of all the variants used in the case studies. It was shown to offer unique features compared to other existing malware visualization systems in the comparative analysis. In the future, RanViz can be expanded to apply this approach to other malware types. Another future development could be creating an early detection system that works with the operating system and uses a machine learning model to detect and halt ransomware execution in real-time based on the initial API call behaviour. This would help in applying the approach used in the development of RanViz to both real-time incidents and post-incident interventions against ransomware.

REFERENCES

[1] P. O'Kane, S. Sezer, and D. Carlin, "Evolution of ransomware," *IET Netw.*, vol. 7, no. 5, pp. 321–327, Jun. 2018.

[2] G. O. Gorman and G. McDonald, "Ransomware : A growing menace," *Symantec*, vol. 1, p. 16, Aug. 2012.

[3] H. Tuttle, "Ransomware attacks pose growing threat," *Risk Manage.*, vol. 63, no. 4, pp. 4–7, 2016.

[4] *Threat of Ransomware Remains at Peak With Half of Organizations Falling Victim in the Last Year*, Athena Inf. Solutions Pvt. Ltd, India, New Delhi, 2023, pp. 1–4.

[5] P. Chakraborty, "Ransomware remains major threat as Sophos reports state of cyber security in 2023," Gurgaon Athena Information Solutions Pvt. Ltd, India, Tech. Rep., 2023, pp. 2023–2024.

[6] M. Sunidhi, "Elastic global threat report 2023 reveals dominance of ransomware," Athena Information Solutions Pvt. Ltd, India, Mumbai, Tech. Rep., 2023, pp. 3–5.

[7] *CISO Research Reveals 90 % of Organisations Suffered at Least One Major Cyber Attack in the Last Year; 83 % Report Ransomware Payments*, Athena Information Solutions Pvt. Ltd, India, Mumbai, 2023, pp. 1–3.

[8] J. Porter, "Wolverine part of massive insomniac games leak after ransomware deadline passes," Verge New York City, USA, Tech. Rep., 2023.

[9] B. Yamany, M. S. Elsayed, A. D. Jurcut, N. Abdelbaki, and M. A. Azer, "A holistic approach to ransomware classification: Leveraging static and dynamic analysis with visualization," *Information*, vol. 15, no. 1, p. 46, Jan. 2024.

[10] Y. Wang, Z. Li, and Y. Zhang, "Optimized ransomware detection through reverse Bayer analysis of file system activities," *OSF Preprints*, 2024.

[11] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: Recent advances, analysis, challenges and future research directions," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102490.

[12] H. N. Nguyen, F. Abri, V. Pham, M. Chatterjee, A. S. Namin, and T. Dang, "MalView: Interactive visual analytics for comprehending malware behavior," *IEEE Access*, vol. 10, pp. 99909–99930, 2022.

[13] M. Alazab, S. Venkataraman, and P. Watters, "Towards understanding malware behaviour by the extraction of API calls," in *Proc. 2nd Cybercrime Trustworthy Comput. Workshop*, Jul. 2010, pp. 52–59.

[14] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," 2016, *arXiv:1609.03020*.

[15] T. Munzner, "Visualization analysis and design- presentation," *Vis. Anal. Design*, vol. 16, pp. 1–3, Aug. 2014.

[16] M. Wagner, A. Rind, N. Thür, and W. Aigner, "A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS," *Comput. Secur.*, vol. 67, pp. 1–15, Jun. 2017.

[17] B. C. M. Cappers, P. N. Meessen, S. Etalle, and J. J. Van Wijk, "Eventpad: Rapid malware analysis and reverse engineering using visual analytics," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2018, pp. 1–8.

[18] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, and W. Aigner, "A survey of visualization systems for malware analysis," in *Proc. Eurograph. Conf. Vis.-State Art Rep., EuroVis-STAR*, Jan. 2015, pp. 105–125.

[19] S. Poudyal, K. P. Subedi, and D. Dasgupta, "A framework for analyzing ransomware using machine learning," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1692–1699.

[20] J. Rafapa and A. Konokix, "Ransomware detection using aggregated random forest technique with recent variants," *Authorea*, pp. 1–8, Aug. 2024.

[21] S. Razaulla, C. Fachkha, C. Markarian, A. Gawanmeh, W. Mansoor, B. C. M. Fung, and C. Assi, "The age of ransomware: A survey on the evolution, taxonomy, and research directions," *IEEE Access*, vol. 11, pp. 40698–40723, 2023.

[22] V. Cobilean, H. S. Mavikumbure, B. J. Mcbride, B. Vaagensmith, V. K. Singh, R. Li, C. Rieger, and M. Manic, "A review of visualization methods for cyber-physical security: Smart grid case study," *IEEE Access*, vol. 11, pp. 59788–59803, 2023.

[23] G. Richer, A. Pister, M. Abdelaal, J.-D. Fekete, M. Sedlmair, and D. Weiskopf, "Scalability in visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 7, pp. 3314–3330, Jul. 2024.

[24] A. Ulmer, M. Angelini, J.-D. Fekete, J. Kohlhammer, and T. May, "A survey on progressive visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 6447–6467, Sep. 2024.

[25] A. Singh, A. Ikuesan, and H. Venter, "A context-aware trigger mechanism for ransomware forensics," in *Proc. 14th Int. Conf. Cyber Warfare Secur. (ICCWS)*, 2019, pp. 629–638.

[26] A. Patel and J. Tailor, "A malicious activity monitoring mechanism to detect and prevent ransomware," *Comput. Fraud Secur.*, vol. 2020, no. 1, pp. 14–19, Jan. 2020.

[27] R. Richardson and M. M. North, "Ransomware: Evolution, mitigation and prevention," *Authorized Administrator Digit. Commons Kennesaw State Univ.*, vol. 13, no. 1, pp. 10–21, 2017.

[28] K. Hammadeh and M. Kavitha, "Unraveling ransomware: Detecting threats with advanced machine learning algorithms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 9, pp. 484–491, 2023.

[29] H. Sistemas. (2019). *Virustotal Public API V2.0.* Accessed: Apr. 7, 2024. [Online]. Available: https://www.virustotal.com/en/documentation/public-api/

[30] *IPStack API*. Accessed: Jul. 5, 2024. [Online]. Available: https://ipstack.com/documentation

[31] *Tshark*. Accessed: Apr. 5, 2024. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[32] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Malicious behavior patterns," in *Proc. IEEE 8th Int. Symp. Service Oriented Syst. Eng.*, Apr. 2014, pp. 384–389.

[33] A. Singh, R. A. Ikuesan, and H. S. Venter, "Ransomware detection using process memory," in *Proc. Int. Conf. Cyber Warfare Secur.*, vol. 17, Mar. 2022, pp. 413–422.

[34] T. R. Dendere and A. Singh, "Ransomware detection using portable executable imports," in *Proc. Int. Conf. Cyber Warfare Secur.*, Mar. 2024, vol. 19, no. 1, pp. 66–74.

[35] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "AI-based ransomware detection: A comprehensive review," *IEEE Access*, vol. 12, pp. 136666–136695, 2024.

[36] Corvus Forensics. (2019). *Virus Share*. Accessed: Apr. 4, 2024. [Online]. Available: https://virusshare.com/

[37] N. Naik, P. Jenkins, N. Savage, L. Yang, T. Boongoen, N. Iam-On, K. Naik, and J. Song, "Embedded YARA rules: Strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis," *Complex Intell. Syst.*, vol. 7, no. 2, pp. 687–702, Apr. 2021.

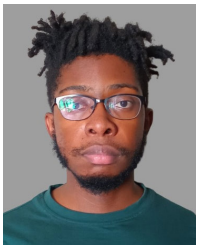
[38] A. Panaras, B. Silverstein, and S. Edwards, "Automated cooperative clustering for proactive ransomware detection and mitigation using machine learning," *TechRxiv*, pp. 1–9, Sep. 2024.

[39] S. Zhang, T. Du, P. Shi, X. Su, and Y. Han, "Early detection and defense countermeasure inference of ransomware based on API sequence," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 10, pp. 632–641, 2023.

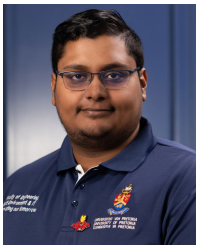
[40] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 5769–5779.

[41] *Python*. Accessed: Apr. 4, 2024. [Online]. Available: https://docs.python.org/

[42] A. Singh and R. A. Ikuesan, "MalFe—Malware feature engineering generation platform," MDPI, Basel, Switzerland, Tech. Rep., 2023, pp. 1–20.
[43] *Django Auth*. Accessed: Sep. 3, 2024. [Online]. Available: https://docs.djangoproject.com/en/2.2/topics/auth/passwords
[44] *Apex Charts*. Accessed: Sep. 3, 2024. [Online]. Available: https://apexcharts.com/docs/installation/
[45] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, 1990, pp. 249–256.
[46] NIST. (2014). *Computer Forensics Tool Testing Program*. Accessed: Oct. 5, 2024. [Online]. Available: https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt
[47] ANY.RUN LLC. *ANY.RUN Interactive Online Malware Sandbox*. Accessed: Oct. 5, 2024. [Online]. Available: https://any.run/
[48] Payload Secur. *Free Automated Malware Analysis Service Hybrid Analysis*. Accessed: Oct. 5, 2024. [Online]. Available: https://www.hybrid-analysis.com/
[49] S. Usharani, P. M. Bala, and M. M. J. Mary, "Dynamic analysis on crypto-ransomware by using machine learning: GandCrab ransomware," *J. Phys., Conf. Ser.*, vol. 1717, no. 1, Jan. 2021, Art. no. 012024.
[50] R. Umar, I. Riadi, and R. S. Kusuma, "Mitigating sodinokibi ransomware attack on cloud network using software-defined networking (SDN)," *Int. J. Saf. Secur. Eng.*, vol. 11, no. 3, pp. 239–246, Jun. 2021.
[51] J. S. Aidan, H. K. Verma, and L. K. Awasthi, "Comprehensive survey on petya ransomware attack," in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst. (ICNGCIS)*, Dec. 2017, pp. 122–125.



VHUHWAVHO MOKOMA received the B.Sc. degree in computer science and computer engineering from the University of Cape Town. He is currently pursuing the B.Sc. degree (Hons.) in computer science with the University of Pretoria. He is also affiliated with the Digital Forensic Science Research (DigiForS) Group, University of Pretoria, which focuses on finding innovative security solutions and addressing the emerging problem areas in digital forensics, distributed trust, and security issues in pervasive computing. His research interests include ransowmware and malware forensics.



AVINASH SINGH received the B.Sc. and M.Sc. degrees (Hons.) in computer science from the University of Pretoria.

He is a dedicated Researcher, specializing in digital forensics, with a focus on ransomware and malware forensics and digital forensic readiness. He is currently a Lecturer with the Department of Computer Science, University of Pretoria. He has actively contributed to the academic community by participating in international conferences and publishing his work in reputable journals. He is affiliated with the prestigious Golden Key International Honour Society and the Digital Forensic Science (DigiForS) Research Group. In addition to his research and academic responsibilities, he leads the Intelligent Cyber Forensics Laboratory (ICFL), University of Pretoria, which focuses on advancing forensic methodologies and technologies with artificial intelligence. His commitment to scholarly excellence extends to peer-reviewing several articles for various academic journals.

Mr. Singh serves as a Review Committee Member for the Information Security South Africa (ISSA) Conference.