2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 979-8-3503-8199-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/TrustCom60117.2023.00221

Apt Detection of Ransomware - An Approach to Detect Advanced Persistent Threats Using System Call Information

Rudra Prasad Baksi
School of Information Technology
Illinois State University
Normal, IL, USA
rpbaksi@ilstu.edu

Vishwas Nalka
Dept. of Computer Science & Engineering
University at Buffalo, SUNY
Buffalo, NY, USA
vishwasn@buffalo.edu

Shambhu Upadhyaya
Dept. of Computer Science & Engineering
University at Buffalo, SUNY
Buffalo, NY, USA
shambhu@buffalo.edu

***Abstract*—Ransomware of the Advanced Persistent Threat (APT) type are very sophisticated and often have a contingency plan of attack in case they are discovered while the attack is in progress. Due to the ever-changing trait of such APT-type ransomware, an intelligent and robust intrusion detection system (IDS) is the need of the hour and in this paper, we put forward machine learning (ML) and natural language processing (NLP) based intrusion detection systems. We utilize a commercial simulator to run different real-world ransomware attacks to create, for the first time, a dataset for APT-type ransomware research. Then, we develop multiple IDSes by training ML models like support vector machine (SVM), logistic regression (LR), gradient boosting (GB) decision trees, random forest (RF), naive Bayes classifier (NBC), and an NLP model called BERT, on this dataset. With our intelligent IDS, we could precisely distinguish the system calls of processes spawned by ransomware from legitimate system calls. We compare the different intrusion detection systems developed using the six aforementioned models. The IDS using the NLP BERT model achieves the best accuracy of 99.98%, and the IDS using the Naive Bayes Classifier achieves an accuracy of 98.55%. Furthermore, we discuss the tradeoffs of these models for designing an intelligent IDS. The advancement in cyber attacks, especially ransomware-based attacks, necessitates this upgrade in IDS which is essential for a strong defense.**

***Index Terms*—Advanced Persistent Threats (APT), Artificial Intelligence (AI), Cybersecurity, Intrusion Detection System (IDS), Machine Learning (ML), Natural Language Processing (NLP), Ransomware.**

I. Introduction

Ransomware are a type of malware which encrypt critical data of a system and hold them for ransom. The data under siege is released by the attacker when they receive the ransom. They generally refrain from releasing the encrypted resources if the ransom is not paid. Some of the recent sophisticated ransomware variants possess a few additional characteristics including existence of a "campaign abort" or a "contingency plan of attack" upon discovery. Such ransomware variants are often termed as advanced persistent threats (APT) and are generally perpetrated by nation state actors with huge amount of resources at their disposal [1].

According to a study by Ponemon Institute, the average financial losses suffered by a company owing to the damaged reputation after an APT attack, amounts to about $9.4 Million [2]. APT ransomware campaigns like WannaCry, Petya, and NotPetya caused considerable financial losses to the victims [1]. According to published reports, between May 12, 2017 and May 17, 2017, WannaCry collected $75,000 to $80,000 in ransom [3], [4]. Off late, the malware WannaCash is also causing trouble to the cyber-world [5]. Recently, Indian nuclear power plants became victims of data breaches [6]. Colonial Pipeline Co., one of the biggest oil and gas pipeline companies in the USA, came under ransomware attack in May 2021 [7]. This attack caused widespread gas shortages, disruptions in functioning of airports and gas stations and caused gas prices to go up. A ransom of $4.4 Million was paid to receive the decryption key [8], [9]. The following characteristics make these ransomware attacks a true APT: 1) exploiting zero-day vulnerabilities to achieve their goal, 2) non-stop campaign until goals are achieved, 3) adaptive and having the ability to attack high value targets through multiple modes of attack [1], [10], and 4) using stealth to quietly invade in a series of steps [11]. The financial losses keep increasing with time, and when the target is a government agency, then national security is put to risk. APT-type ransomware attacks are causing trouble not only to the government organizations but also to the industrial systems and other organizations and/or institutions, which directly affect daily lives of the masses.

The aforementioned factors and incidents outline a great threat to the critical infrastructure as a whole, be it government or industry. The problems are intense and the attacks are adaptive in nature, requiring a holistic approach to address them. With more emerging threats, the attack dynamic is rapidly changing. These sophisticated ransomware stealthily infiltrate the system and mount the attack through multiple stages. They disguise the processes as legitimate processes in the system. This warrants the introduction of a smarter IDS rather than rule-based or signature-based ones [12], [13]. In this paper, we explore multiple models to make an IDS smarter with the use of machine learning (ML) and natural language processing (NLP). Using a commercial tool, we

2324-9013/23/$31.00 ©2023 IEEE
DOI 10.1109/TrustCom60117.2023.00221

create a dataset consisting of system call information obtained by running a few APT-type ransomware, viz. BlackByte, BlackMatter, Diavol, REvil, and Darkside. Then we develop IDSes by training the ML and NLP-based detection models on this dataset and compare their accuracy and performance. The major contributions of this paper are the creation of a dataset from APT-type ransomware and the development of ML and NLP-based intelligent detection models. Since we utilize a modern industry-grade commercial tool to simulate the ransomware and create a dataset from the system calls and other metadata, the dataset we generate is a close representation of real-world attack data. This can be beneficial in designing and testing any new detection techniques. The paper is organized as follows. In Section II, we provide the background on ransomware and discuss relevant research work on their detection. Following this, in Section III, we put forward our intelligent and apt intrusion detection system (IDS). In Section IV, we present the details of the experimental setup and the dataset creation. Then, in Section V, we discuss the effectiveness of the various detection models and their performance. Finally, in Section VI, we conclude the paper and discuss the future prospects of our research.

II. Background and Related Work

APT groups mount attacks through multiple stages by creating sophisticated malware. Lockheed Martin's "Cyber Kill Chain" framework describes an APT through a seven-stage life cycle [14]. LogRythm describes an APT through a five-stage life cycle [15]. Baksi and Upadhyaya [1] describe APT through a set of five characteristics exhibited by sophisticated malware. The detection of sophisticated APT malware in the ever-changing attack landscape is a daunting task. People have explored quite a few options for this purpose. One such approach is security incident and event management (SIEM), and SIEM-like detection mechanism [16]. Milajerdi et al. [17] used a graph-based technique. Here, they create an audit log and thereafter, using a causality tracker, they generate a provenance graph. This graph is then passed through a policy matching engine and noise filter to create a high-level scenario graph (HSG). This HSG is used by a detection engine to detect any APT malware present in the system. Their goal is achieved through alert generation, alert correlation, and attack scenario presentation. Contrary to this, in our paper we use ubiquitous system call logs to look for any intrusion in the system. We use AI models to train, test, and validate on the dataset created using system call logs. These models are then used to detect the APT malware. The research presented by Liu et al. [18] uses both graph-based techniques as well as AI techniques. Their solution involves a heuristic approach that converts log entries into a heterogeneous graph. Then each graph is transformed into low-dimension vectors using the *log2vec* technique which involves the usage of word2vec processes. They managed to detect both malicious and benign types of APT threats. Their detection algorithm comprises of a clustering algorithm, a threshold detector, and selection of parameters for the detection algorithm. Besides APTs, they also detect insider threats using their log2vec technique. They show that log2vec outperforms deep learning and Hidden Markov Model (HMM) based models. But their solutions have limitations, which includes redefining the graph rules for different processes, high false positive rates (FPR), and the task of choosing parameters for the detection algorithm. In contrast to their approach, our solution uses NLP and ML based models to detect a particular type of malware, viz., APT-type ransomware. In our technique, the *rules of engagement* are not required to be redefined for different processes and/or different malware.

Ransomware can be categorized into three principal categories, namely *the locker*, *the crypto* and *the hybrid* [19]. The threat model, i.e., the ransomware considered in this paper is of the APT kind. They have the capabilities of a generic ransomware as well as that of an APT-type malware. They are often sophisticated enough to have a contingency plan of attack on being discovered [1], henceforth called APT ransomware. BlackByte, an APT ransomware, has two modes of attack. It can either attack directly or offer services as ransomware-as-a-service (RaaS) [20]. It exploits the ProxyShell vulnerabilities that exist in Microsoft Exchange Server to infiltrate the system. The ransomware has a Russian origin, as it avoids any devices that have language settings in Russian and/or some other language from any of the former Soviet countries. The primary targets include US-based organizations in critical infrastructure sectors such as government, finance, and food & agriculture [21]. DarkSide is another APT ransomware that was involved in attacks on Colonial Pipelines and Toshiba [9]. REvil is also an APT ransomware which was responsible for attacks on entities that are suppliers of Apple Inc. and are responsible for stealing confidential information [22]. Both REvil and DarkSide have similar code bases. Just like BlackByte, both REvil and DarkSide allegedly have Russian origin. They avoid devices that have language settings similar to Russian and former Soviet countries [23]. BlackMatter is another APT ransomware, which targeted multiple U.S. critical infrastructure entities, including two U.S. Food and Agriculture Sector organizations [24]. They are allegedly a "rebrand" of DarkSide ransomware and their main targets include the food and agricultural sector [25]. Diavol, an APT ransomware, is supposedly linked to a cybercrime group called Wizard Spider who are also known as Trickbot. They are also the perpetrators of the ransomware Ryuk, Conti and the spam trojan Emotet [26]. Diavol, just like the other ransomware created by TrickBot, attacks corporate entities, especially financial institutions which used Windows [27], [28]. TrickBot is a nation-state actor with apparent connections to Russian intelligence agencies [29].

In this paper, we recreate the attacks of the five aforementioned ransomware via simulation using a commercial tool and create a dataset out of it. Then, we develop classifier-based IDS using this new dataset to detect APT ransomware. Our objective is to classify a system call as malicious or benign, and this is a binary classification task. We develop the IDS based on this classification problem, and we consider

two kinds of IDS in our work. One is based on machine learning, and the other is based on transformer architecture (neural networks). The first kind is machine learning-based IDS, for which we choose five machine learning algorithms to develop different IDSes independently. The second kind of IDS is NLP-based, designed using the BERT model (which uses a transformer neural network). Further, we evaluate these six types of IDSes and provide a comparison between them.

A. Machine Learning Algorithms

In this section, we provide a brief overview of the machine learning algorithms used in the design of the IDS. We chose five algorithms in our experiment, viz. Naive Bayes, Support Vector Machine, Logistic Regression, Gradient Boosting Decision Trees, and Random Forest algorithms. These are the models that are used extensively by researchers for binary classification. Naive Bayes [30] is a supervised machine learning algorithm based on applying Bayes' theorem along with the conditional independence assumption between every pair of features. The Naive Bayes Classifier can work well on categorical data and can be extended to text-based classification. Support Vector Machine [31] is a type of supervised learning method used for classification, regression, and outlier detection. The SVM classification algorithm has been widely utilized in various applications such as email spam filtering and developing IDS. Logistic Regression [32] is another supervised learning model to perform binary or multivariate classification. The Logistic Regression model is one of the first choices for binary classification. The other approaches that give good results are ensemble models, where multiple models are combined to improve the classification performance. Random Forests [33], and Gradient Boosting Decision Trees [34] are two powerful algorithms, that are based on the ensemble learning method, considered in our IDS design. In our research, we used the aforementioned models and compared the results using standard metrics such as accuracy, F1 score, and training time [35] to learn how they performed.

B. Datasets and ML-based IDS

Researchers have long used the KDD CUP'99 dataset to design and evaluate IDSes [36]. Using this dataset, researchers can design classifiers working on 41 features to detect attacks like denial of service (DoS), probe, remote to local (R2L), and user to root (U2R) [37]. NSL KDD is a more refined version of the KDD CUP'99 dataset by removing several of its integral issues and is used for the detection of attacks similar to the KDD CUP'99 dataset [38], [39]. These datasets are mostly used for designing network-based IDS (NIDS) [37]. On the other hand, researchers have been plagued by the paucity of data regarding APT ransomware. To overcome this problem, in our research, we create a dataset consisting of system call logs of APT ransomware. We believe the research community can utilize this dataset to design and evaluate host-based IDS (HIDS) in the context of APT-type ransomware attacks. This dataset is discussed further in Section IV.

Traditionally, machine Learning has been used to develop IDS. In [40], Kruegel and Toth show how the machine learning model Decision Trees was employed to improve a signature-based intrusion detection system. Our work focuses on utilizing ML/AI on the system calls to directly identify intrusion as opposed to rule-based or signature-based methods, which are dynamic and constantly keep changing. Hamza et al. [41] developed a system that can translate Manufacturer Usage Description (MUD) policies into flow rules using Software-defined networking (SDN), and then used for designing IDSes in the IoT network. Jamshed et al. [42] presented Kargus, a highly-scalable software-based NIDS compatible with Snort. In contrast, we worked on developing an AI-based host-based IDS (HIDS) that can be utilized in a system directly to detect intrusion by APT ransomware. Furthermore, we provide a comparison of various ML/AI models and give the users a choice based on the computation power available to them and the trade-offs between the best-performing models.

Alkasassbeh et al. [43] used data mining techniques along with classification techniques like multi-layer perceptron (MLP), Naive Bayes' classifier (NBC), and Random Forest to detect distributed denial of service (DDoS) attack. Almaseidin et al. [35] used classifiers like J48, Random Forest, Random Tree, Decision Table, MLP, NBC, and Bayes Network on the KDD dataset to design and evaluate IDSes to detect attacks like DoS, U2R, R2L, and Probe. Halimaa et al. [44] used SVM, and NBC on NSL-KDD dataset to detect DoS, Probe and R2L attacks. Keserwani et al. [45] used a combination of Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) to obtain features from the dataset. They used random forest (RF) for the purpose of attack detection. Krishna and Arunkumar [46] also proposed a hybrid GWO-PSO optimization algorithm used in conjunction with an RF classifier on the NSL-KDD dataset to detect DoS, R2L, U2R, and Probe attacks. But the existing aforementioned datasets are generally used for designing NIDSes and are not useful for our research since they do not truly represent the attacks that we are trying to address. Therefore, in our research, we created a new dataset for APT ransomware and designed a HIDS to detect the same.

C. Natural Language Processing based Models

Natural Language Processing (NLP) has greatly improved the abstract understanding and representation of language. Earlier, Recurrent Neural Networks (RNN) were used for the purpose, but now transformer based architectures [47] are well-suited for the same. Bidirectional Encoder Representations from Transformers (BERT) [48] is a transformer-based language model. BERT model differs from the traditional transformer architecture in that it uses only the encoder instead of the encoder-decoder design.

Researchers have previously worked on the intersection of NLP and Cybersecurity. Rahali and Akhloufi [49] used transformer-based models to automatically identify malicious software. Andronio et al. [50] presented an NLP-based ransomware detection model called HelDroid for the mobile

device platform. Continella et al. [51] put forward a detection model "ShieldFS", which focuses on the low-level behavior of the ransomware. Tran and Sato used an NLP-based approach to analyze and classify malware from the data collected from API call sequences [52]. They collected behavioral data from malware from API call sequences. Thereafter, they performed feature extractions and feature vectorization using TF-IDF and Paragraph Vectors. Following that, classification was done using KNN, SVM, RF, and MLP. Najafi et al. [53] used the events log of SIEM at enterprise systems to model the behavior as directed acyclic graphs (DAG) and then used NLP-based approach to detect graph-based outliers. Wang et al. [54] presented research on the challenging problem of APT attribution. In their research, for the intermediate representations, they used the VEX IR method. They chose two features, viz. string features and code features. They applied a random forest classifier (RFC) and deep neural network (DNN) classifier to learn and classify. Finally, they ran RFC and Local Interpretable Model-agnostic Explanations (LIME) to interpret the results from the classification task.

In our research, we posit that any malicious process that can disguise itself would eventually execute a devious command, such as deleting a shadow copy or running a standard encryption call. This will allow us to detect and identify the ransomware in the system. This implies that the system call logs and other process metadata can be useful features that can be used to identify an intrusion in the system. We have devised a way to identify APT ransomware using modern NLP techniques and compared this with traditional machine learning classifier-based approaches. The focus is on using system calls and other metadata captured about the processes, to generate a dataset that allows us to design a classifier-based IDS to detect ransomware that disguises itself as legitimate processes.

III. The APT Detection System

Intrusion Detection Systems (IDS) are responsible for identifying various forms of infiltration in the system with malicious intent. In this paper, we design an intelligent IDS to detect intrusions by identifying malicious processes. Our intelligent IDS is trained on the system call information during the ransomware attack simulation. This allows the IDS to learn the malicious calls (attack patterns) in the system and can effectively discern the benign and malicious system calls in the test dataset (the data which is unseen by the IDS during model training). This is analogous to being able to identify rogue system calls during the actual attack using the previously trained intelligent IDS. Microsoft Windows systems capture various metadata about a process that is running such as process name, description, command line system call executed, operation performed, and so on. Analyzing this metadata can give critical insights into determining whether the process is malicious or not. Analyzing the metadata using ML and NLP models will help in the detection of malicious processes in the system which stealthily disguise themselves as legitimate processes. This can be viewed as a classification task where the system call metadata can be classified into "Malicious" or "Non-Malicious." The IDS developed in this paper uses the aforementioned approach of using the system call metadata to differentiate between malicious and non-malicious processes. We use a few ML models and one NLP model for the classification task. The IDS has been designed to identify APT ransomware.

The designing of AI based IDSes requires a dataset to be used for the purpose of training, validation, and testing. Since existing datasets are not suitable for our study as described before, we made use of a commercial simulator from Picus Security [55] to generate our own data. The simulator provided us with real-world APT ransomware and we ran it inside the Windows sandbox environment. We then collected the system call metadata and created the dataset to be used for designing the IDSes.

The ML-based IDS was designed using multiple ML models and the results were compared. The models used were NBC, LR, RF, SVM, and GB decision trees. These models were trained using the dataset created from the simulator. The TF-IDF based vectorization was used to feed the data for the Naive Bayes Classifier and the word embeddings based vectorization was used to train the models SVM, RF, GB decision tree, and LR.

The NLP-based IDS was designed using a BERT model for sequence classification. This transformer model contains the bare BERT Model architecture with a linear layer on top of the pooled output layer. This linear layer can be trained for sequence classification. The BERT Model is loaded from the pre-trained model configuration that is available open-source. The weights/parameters for the bare BERT Model architecture are instantiated from the pre-trained model and the linear layer on the top can be fine-tuned and the weights can be trained with the dataset available for the classification task.

Figure 1 provides a summary of the various machine learning classifiers and the NLP model used in the IDS design. The NLP-based BERT classification model is compared with the machine learning models – Naive Bayes, Support Vector Machine, Logistic Regression, Random Forest, and Gradient Boosting Decision Trees, and the performance of these models is benchmarked using standard metrics.

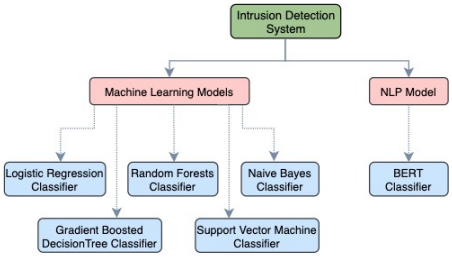

Fig. 1. Models evaluated for the IDS design

IV. Experimental Setup

The procedure to generate data, run simulations, and develop the models consists of the following steps:

1) Ransomware attack simulation in a sandbox environment.
2) Capture system calls, metadata of non-malicious processes and the system calls during the ransomware execution.
3) Create the dataset from the system calls and process metadata dumps.
4) Develop the classifier models for the classification of malicious/non-malicious calls.
5) Evaluate and compare the models.

Figure 2 gives a pictorial view of the experimental setup starting with the ransomware simulation to the final step of model evaluation. Figure 3 illustrates the various steps in each phase of the experiment workflow from generation of the dataset to model evaluation.

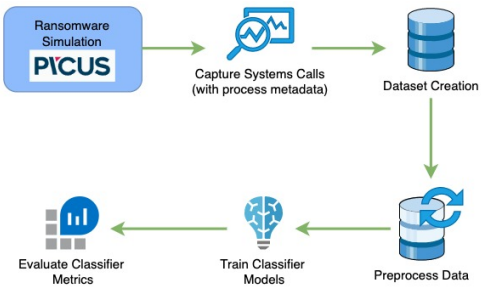

Fig. 2. Experimental Setup

A. Ransomware & Simulated Environment

In our research, we used the simulator provided to us by Picus Security. The platform contains various attacks which denote the different stages in the Unified Kill Chain and contain multiple attack scenarios as defined in the MITRE ATT&CK® tactics [56], [57]. In our controlled experiment inside the sandbox, we used five prominent and representative APT ransomware, viz. BlackByte, BlackMatter, Diavol, Darkside, and REvil. The simulation was set up in a Dell OptiPlex 7010 system with Intel core i7-3770 @ 3.40GHz processor, 16.00 GB memory running a 64-bit Windows 10 Education OS, and Intel HD Graphics 4000. The simulations were run in a safe sandbox environment. We used Windows Sandbox application feature (available from Windows 10). Windows Sandbox is a lightweight environment for desktop intended for safely running software in isolation. The configuration of spawned Windows Sandbox was Intel core i7-3770 @ 3.40GHz processor, 4.00 GB RAM running a 64-bit Windows 10 Enterprise OS. The Picus platform requires the installation of an agent, which is used to run the simulations. This agent was installed in the Windows Sandbox and all simulations were run inside of it. After the agent is installed, the platform aids in simulating real-world attacks and threats against our system. It can simulate various exploits and attacks that operationalize the frameworks such as MITRE ATT&CK® techniques. Furthermore, the ransomware simulations are made of independent adversary techniques but these scenarios (exploits, attacks) do not run any malicious code and only notepad-like ”safe apps to prove code execution. It does not actually lock a user out of a system or encrypt the entire system and ask for a ransom to regain access. It is like a scaled down version of the ransomware that can still execute and simulate the attacks or exploits but in a nondestructive fashion. This allows us to capture enough data on the various kinds of actions performed by a ransomware such as encrypting the drives (files, volumes) or deleting the shadow copies or unblocking access to files and deleting them from the system. After each attack simulation, the Picus agent follows up with clean-up functions to restore and clean-up the OS to the last known state. The creation of the dataset and preprocessing of the data are described in the next section.

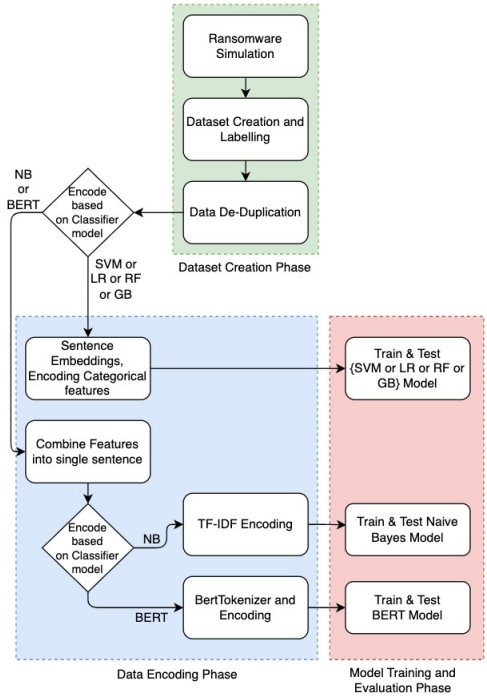

Fig. 3. Experiment Workflow

B. Dataset

Windows Sysinternals offers a tool called Process Monitor (Procmon). It is an advanced monitoring tool, that shows real-time file system, registry, and process/thread activity. It can capture reliable process data such as process name, command line system calls executed, user ID, operations performed, description, company name, and many more. It can capture these details for every process that is running in the system even if it is spawned in the background. This extensive metadata information can be dumped into a csv file. When

a ransomware is running, it executes multiple malicious processes that perform actions such as encrypting a file, deleting shadow copies, deleting files, and so on. Procmon is used to capture a snapshot of the system while simulation is running. This contains all the metadata about the processes running during the ransomware simulations. Similarly, to capture the non-malicious and default system process events, Procmon is used when the attack simulation is not triggered. This captures all the processes that run in the system in the background and user actions are performed to capture the normal state of the system. This involves performing few user actions such as copying files, creating and deleting text files, opening few applications such as notepad and browser. These Procmon event details are converted into csv files and stored.

The process events captured during ransomware simulations and the ones captured without the ransomware attacks are used to create the dataset for our classification tasks. The features of interest in these events are the Process Name, Operation, Detail, Description, Command Line, and Company. The events captured with no ransomware attacks are de-duplicated and labelled as 'System', i.e., non-malicious and this dataset is referred to as System dataset. The events captured during the ransomware simulation cannot be marked entirely as 'Malicious' as background Windows processes and other non-malicious processes would still be running. Hence, the events captured in the System dataset are referenced to identify the malicious processes. Every event in the data captured during ransomware simulation is compared with the events captured for the System dataset. Based on the uniqueness of the Process Name, Command Line call executed, Operation and Company features, the events are tagged as 'Malicious.' For instance, if all of these features are already available in the System dataset, then that event is ignored since it denotes that it is a background system event. If any of these four features are not already available in the System events dataset, then that particular event is tagged as Malicious. The Detail and Description features are not taken into account while this tagging is performed. If any process event has Detail and Description values which are not there in the System dataset, they may or may not be malicious in nature. If the four features Process Name, Operation, Command Line, and Company denote a process event as safe then those events, even with different Detail and Description features, can be labelled as non-malicious in nature.

The dataset is tagged as 'System' and 'Malicious' as explained above. It is used to build the classification model to identify malicious process events. The dataset created contains 7 columns, viz. Process Name, Operation, Company, Detail, Command Line, Description, and Label. It contains a total of 90,841 rows with 43,487 events tagged as 'Malicious' and 47,354 events tagged as 'System.' In the following subsection, we elucidate the pre-processing of the data to make it suitable for the training of the AI models. As a part of our contribution, we would like to release the dataset for the research community to help researchers working with APT ransomware.

Figure 4 shows a sample of the created dataset with four rows. The first two rows show non-malicious System processes that were captured when no ransomware was run in the Windows sandbox. The subsequent two rows show Malicious processes that were captured as part of the ransomware simulation in the sandbox. The first row depicts the notepad++.exe application running in the system. The data row captured is a thread create event triggered by the notepad++ application process. The other columns of that row show the metadata related to this process. The second row depicts a Windows Explorer.exe application running in the system. An event related to file write is captured and this is a benign data row as well. The third row represents a malicious process running in the system. This was triggered as part of ransomware and it is using the powershell.exe application to carry out an unwanted operation in the system. The shadow copies are storage extents that are duplicate copies of the original volumes and can be used for back-up/restore in case of system failures. Using the powershell.exe application, the ransomware is trying to delete the shadow copy of the volumes in the system. We can see in the command line that it is using a combination of Get-CimInstance and Remove-CimInstance scripting tools of the powershell.exe to carry out its malicious task. The last row of Figure 4 represents a malicious process named BlackByteEncryptor.exe which is part of the BlackByte ransomware campaign. As the name suggests, this is a dangerous process that is trying to encrypt the files in the system. Encrypting the files, volumes in system is a common technique used by ransomware to lock the user out of the system and prevent access to the drives (system storage). The command line columns of the fourth row shows how this encryptor process is trying to encrypt a text file available in a certain location.

C. Data Preprocessing

The dataset contains seven columns as depicted in Figure 4 with six columns for features and one column representing the label. Each of these six features is text-based data. The columns Process Name and Operation are categorical features, and the columns Detail, Company, Command Line, and Description are text data. These text-based features cannot be used directly for training the models. Instead, they need to be converted into numerical features. The categorical features are converted into one-hot encoding and the text data columns are converted into vectors of word embedding. To get the word embedding, pretrained Stanford Glove word embeddings are used [58]. The pretrained word embeddings provide vector representation for several words. The package glove.6B.300d, which contains pre-trained word vectors with 6B tokens, and 400k vocab with a vector of size 300 was used. The Out-Of-Vocab words (any word for which vector representation is not available in the package) are initialized to a random vector of dimension 300 and stored so that the same vectors are used for a word. The sentences are converted to vectors by average pooling the word embedding of each word in the sentence to get a final sentence embedding. Finally, to reduce the number of dimensions, the sentence vector of each of the

Process Name	Operation	Detail	Company	Description	Command Line	label
notepad++.exe	Thread Create	Thread ID: 2440	Don HO don.h@free.fr	Notepad++ : a free (GPL) source code editor	"C:\Program Files\Notepad++\notepad++.exe"	System
Explorer.EXE	WriteFile	Offset: 4,366, Length: 242	Microsoft Corporation	Windows Explorer	C:\Windows\Explorer.EXE	System
powershell.exe	RegEnumKey	Index: 9, Name: {CF78C6DE-64A2-4799-B506-89ADFF5D16D6}	Microsoft Corporation	Windows PowerShell	"powershell.exe" -c "Get-CimInstance Win32_ShadowCopy | Remove-CimInstance"	Malicious
BlackByteEncryptor.exe	RegQueryValue	Type: REG_SZ, Length: 8, Data: 2.0		BlackByteRansomware Encryption	C:\Users\WDAGUtilityAccount\AppData\Local\Temp\BlackByteEncryptor.exe http://pcsdl.com/short-url-v2/000917988986/forest___e486af8e-1d93-4a7e-a9a0-87b94b0ca71b.png C:\Users\WDAGUtilityAccount\AppData\Local\Temp\test.txt	Malicious

Fig. 4. Dataset Sample

features is again average pooled to get one final vector of size 300 that represents the four columns. The one-hot encodings of the categorical columns are merged and combined into one numerical array and this is further extended with the vector representation of the four text columns to get a numerical array of dimension 457 that represents one event in the dataset. This is used to train SVM, Random Forests, Gradient Boosting Decision Trees, and Logistic Regression models.

The dataset is processed in a different manner for the Naive Bayes model. The features are joined by whitespace into a single text sequence and are converted into a meaningful representation of numbers based on the TF-IDF (Term Frequency Inverse Document Frequency) vectorization method. This is a method that takes into account the relevance and importance of words in a corpus (collection of documents which is the collection of all text sequences in this case). The relevance and importance of words are derived from the occurrence count of that word in a document and the number of documents in which the word occurs. Each text sequence is converted into a vector of length 17,125 (this is the number of unique tokens in the document corpus).

In the case of the BERT model, the input is expected to be a text sequence. Since each of the features is of type text data, all the features are joined by whitespace into one text sequence which is fed to the BERT-based tokenizer for encoding. The BERT tokenizer can then use this text sequence to extract word pieces that are encoded into numerical values and can be fed into the model.

D. Model Training and Evaluation

The IDS needs to perform a binary classification to identify a system call along with other metadata as malicious or non-malicious. Since this is a binary classification task, models such as Naive Bayes, Support Vector Machines, Logistic Regression, Random Forests, and Gradient Boosting Decision Trees classifiers are trained using the dataset. We also train the NLP-based BERT model on the same dataset. We then compare the results from all the aforementioned AI models.

The preprocessed dataset is split into training, validation, and test sets in the ratio 7:1:2 respectively. This means 70% of the data is used to train the model, 10% of the data is used for validation, and the remaining 20% is used for testing and computing the accuracy metrics of the models.

For the ML models, the entire training data is introduced during the training stage. The validation and test sets are used to evaluate the model after the training is completed. These models are trained and evaluated on an MacBook Pro with M1 chip processor and 16GB memory. The scikit-learn package was used to implement the machine learning classifiers. For Naive Bayes classifier, MultinomialNB (multinomial Naive Bayes) model was used with alpha (smoothing parameter) set to 1.0. The logistic regression classifier LogisticRegression is used with the following parameters – penalty (the norm of the penalty term) used was 'l2', the C parameter (inverse of the regularization strength) is set to 1.0, class_weight parameter (weights associated with the classes) was set to 'balanced' mode, the solver parameter (algorithm used in the optimization problem) used was 'newton-cg', and the max_iter value (maximum number of iteration to be used for convergence) is set to 1000. The class SVC (support vector classification) is used for the support vector machine model. The best performing model was obtained with the following parameters – the C parameter (regularization parameter) is set to 10, the kernel parameter is set to 'rbf', the gamma parameter (kernel coefficient) is set to 'auto' (this implies the algorithm will use 1/n_features as the coefficient value). The random forest model RandomForestClassifier was used with the following parameters – n_estimators parameter (number of trees in the forest) is set to 100, the criterion parameter (the function to measure how the split happens and its quality) used was 'gini', and the max_features parameter (number of features to look for the best split) is set to 'sqrt'. Finally, the gradient boosted decision tree model GradientBoostingClassifier was used with the following parameters – loss parameter (the loss function to be optimized) was 'log_loss', the learning_rate (the rate at which the contribution of each tree shrinks) is set to 0.1, n_estimators parameters (number of boosting stages to perform) is 100, the criterion parameter (function to capture the quality of the split) used was 'friedman_mse', and the max_depth (max depth of individual estimators) is set to 3.

For the NLP-based BERT model, a similar split of the dataset is done with 70%, 10%, and 20% of the data for training, validation, and testing, respectively. The BERT-based model for sequence classification BERTForSequenceClassification is used for the task. This model is a slightly modified

version of the BERT base model with a linear layer over the pooled output layer that can be fine-tuned and trained for the classification task. The BERT base model is loaded from a pretrained model that is available open-sourced "bert-base-uncased." This is a pretrained model over the English language (Wikipedia and Google BooksCorpus) and it is a case insensitive model. This pretrained model can be used to load the models' weights, parameters, and configuration for the BERT base model encoder. The linear layer on the top intended for classification is loaded with random weights. The entire model (the embeddings layer, the encoder layers, and the linear layer on top) is tuned and the weights are updated as part of the model training step with the created dataset in hand. As part of the training step, the data is fed to the model in batches of 8 and the model was trained for 4 epochs. The BERT model was trained and evaluated in Google Colab using a GPU. The Nvidia Tesla T4 was available in Google Colab for training and testing the BERT model.

The trained models are then evaluated against the validation and test dataset. Both of these datasets are unseen by the model before this step and each event in these datasets is classified as malicious or non-malicious using the models. The classification performed using the model is compared with the actual labels of these events. The predicted labels and the actual labels are assessed to get the evaluation metrics that allow us to compare and evaluate the models.

V. Results and Discussion

A. Results

After performing the experiments and generating the predictions on the test dataset of size 18,169 records, we now compare the results on the basis of F1 score, Accuracy, and the ROC plots.

TABLE I
Model Accuracy and F1 Score

Classifier	Accuracy (%)	F1 Score
BERT	99.98	1.00
Naive Bayes	98.55	0.984
Gradient Boosted DT	97.21	0.971
Random Forest	96.04	0.958
Logistic Regression	94.47	0.943
SVM	74.51	0.694

Table I summarizes the performance of each classifier in terms of accuracy and F1 scores. We can see that the BERT classifier has the highest accuracy and F1 score followed closely by the Naive Bayes classifier. These are followed by Gradient Boosting Decision Tree, Random Forests and Logistic Regression classifiers in the order of decreasing accuracy. The Support Vector Machine classifier has the least accuracy and F1 score.

Figure 5 shows the receiver operating characteristic (ROC) curves plotted for each model. The area under the curve (AUC) computed for BERT is 1 indicating that is able to distinguish between the two classes clearly. The AUC values for Naive Bayes (NB) and Gradient Boosting (GB) Decision

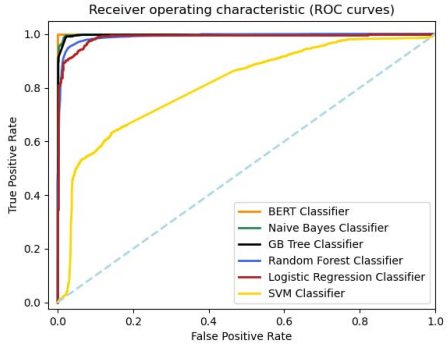

Fig. 5. ROC curves for the Classifiers

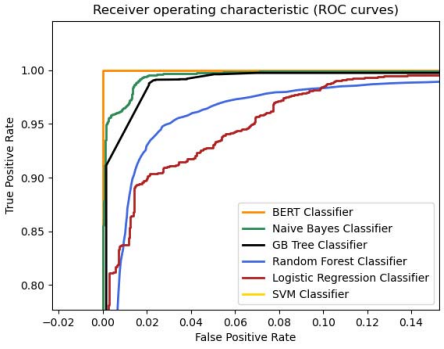

Fig. 6. Zoomed view of the left Top part of ROC curves

Trees are almost equal to 1. This follows a similar trend to the accuracy values and from Figure 6, we could see that, though Naive Bayes model and GB Decision Tree model curves seem similar, the green curve representing Naive Bayes covers more area than the black curve representing GB decision tree. These models are followed by Random Forest and Logistic Regression models in decreasing values of AUC. The support vector machine classifier represented by the yellow curve in Figure 5 has the least area under the curve denoting its poor classification performance on this dataset.

Of all the classification models used, NLP-based BERT model has the best performance, being able to predict the labels for the test set with a near perfect accuracy. The next best performing model is the Naive Bayes Classifier. This is followed by Gradient Boosting Decision Tree, Random Forest Classifier and Logistic Regression Classifier in the order of decreasing accuracy. The model with the least scores is the SVM classifier. On comparing the models with respect to the training time taken as shown in Table II, we can observe a contrast with BERT taking the longest time for training. BERT model took over six hours for four epochs of training. This is followed by SVM classifier which took 5,190.30 seconds, Naive Bayes Classifier took 21.92 seconds, Gradient Boosting Decision Tree took 628.84 seconds, Random Forest Classifier

took 98.65 seconds and Logistic Regression took 6.7 seconds.

B. Discussion

Our experiment and analysis demonstrate the capability of machine learning and NLP-based techniques to develop accurate IDS that can detect APT ransomware. The results show that BERT model and Naive Bayes classifier had high accuracy and are well suited for this use-case. BERT model has the highest accuracy with a perfect F1 score of 1.0 and accuracy of 99.98%. However, the downside of BERT is the high computation requirement with training time of over 6 hours. On the other hand, Naive Bayes Classifier has a slightly lower F1 score of 0.984 and an accuracy of 98.55%. Furthermore, the NB classifier requires a very low training time of around 22 seconds. In situations where accuracy is crucial and the IDS achieving correct classification is important, NLP-based BERT model can be utilized at the expense of longer training time. However, in situations where we cannot afford such intensive computation, training time, and an IDS needs to be trained and run swiftly, the Naive Bayes Classifier can be used at the cost of marginally lower accuracy.

The results illustrate that the NLP-based BERT model performs exceptionally well when text-based data is involved. It is the case with the dataset we created from the system call information, which consists of the process name, operation, detail, company, description, and the command executed in the command line. The downside of the BERT model, as discussed earlier, is its high computation power requirement. Even with a GPU, the training time is significantly higher than the machine learning models. In situations where one cannot afford even a slight dip in accuracy, such as IDS in systems for critical pipeline infrastructure or nuclear power plants, it would be beneficial to use the BERT model that gives the highest accuracy. In contrast, in situations such as shared virtual machines (VMs) in an enterprise, having a dynamic process environment due to new applications or test code constantly installed and updated, one would require frequent retraining of the model to capture non-malicious programs. In such cases, where one can afford a marginally less accuracy but require constant retraining of the models, it would be beneficial to use the machine learning model (Naive Bayes) that can be trained quickly.

TABLE II
CLASSIFIER MODELS AND THEIR TRAINING TIME

Classifier	Training Time
BERT	6 hrs
Support Vector Machine (SVM)	5190.303s
Gradient Boosted Decision Tree	628.84s
Random Forest	98.65s
Naive Bayes Classifier (NBC)	21.92s
Logistic Regression	6.7s

VI. CONCLUSION AND FUTURE WORK

The usage of ML/AI in designing host-based intrusion detection systems demonstrates the intriguing advancement for the apt detection of APT ransomware. In this paper, we have created a dataset using system call information that represents APT ransomware attacks. This dataset can be utilized by researchers in future work to develop intrusion detection systems and carry out research on APT ransomware. We demonstrated the capability of classifier-based IDS, using different classifier models that are machine learning-based and NLP-based. Furthermore, we compared the models on various evaluation metrics and discussed the trade-offs of choosing a model. We provided a choice for the user to utilize one of the above ML/AI models for intrusion detection. The user can make the decision based on their requirements. If the user wants a very high accuracy irrespective of the training time, they can choose BERT. Whereas, if the user requires a faster training time with reasonably good accuracy, they can go for the Naive Bayes Classifier. Though we considered the design and development of IDSes to address the need of the hour (APT ransomware), the approach taken here can be suitably altered to handle generic malware as long as the applications lend themselves to be characterized by system call sequences.

For future work, we like to expand our research into designing classifier-based IDSes with lesser computational complexities and training time. Another aspect of research extending the current work post intrusion detection would be to identify the exact processes or script and their path in the system that execute the malicious commands. Also, it will be interesting to compare the NLP-based BERT model to other deep neural network-based models such as recurrent neural networks (RNN), long short-term memory networks (LSTM), and evaluate the models based on accuracy, computation cost, and ease of implementation.

ACKNOWLEDGMENT

This research is supported in part by the National Science Foundation under Grant No. DGE–2234945. Usual disclaimers apply.

The authors would like to thank Picus Security for providing access to their threat emulation platform. The dataset will be made available by our research group with proper authorization to facilitate further research on APT ransomware.

REFERENCES

[1] R. P. Baksi and S. J. Upadhyaya, "A comprehensive model for elucidating advanced persistent threats (APT)," in *Proceedings of the International Conference on Security and Management (SAM)*, pp. 245–251, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018.

[2] P. I. LLC, "The state of advanced persistent threats," *Ponemon Institute Research Report*, December 2013.

[3] Z. Clark, "The worm that spreads wanacrypt0r," *Malwarebytes Labs*, May 2017.

[4] Secureworks, "Wcry ransomware campaign," *Secureworks Inc.*, May 2017.

[5] T. Meskauskas, "How to uninstall wannacash ncov ransomware?," *PC Risk*, 04 2020.

[6] M. Robbins, "Cyberattack hits indian nuclear plant," *Arms Control Association*, 12 2019.

[7] W. Tourton and K. Mehrotra, "Hackers breached colonial pipeline using compromised password," *Bloomberg*, 06 2021.

[8] C. Eaton and D. Volz, "Colonial pipeline ceo tells why he paid hackers a $4.4 million ransom," *The Wall Street Journal*, 05 2021.

[9] T. M. Research, "What we know about the darkside ransomware and the us pipeline attack," *Trend Micro*, 05 2021.

[10] R. P. Baksi and S. J. Upadhyaya, "Decepticon: a theoretical framework to counter advanced persistent threats," *Information Systems Frontiers*, pp. 1–17, 2020.

[11] R. Mehresh, "Schemes for surviving advanced persistent threats," *Faculty of the Graduate School of the University at Buffalo, State University of New York*, 2013.

[12] V. Kumar and O. P. Sangwan, "Signature based intrusion detection system using snort," *International Journal of Computer Applications & Information Technology*, vol. 1, no. 3, pp. 35–41, 2012.

[13] W. Stallings, *Cryptography and Network Security: Principles and Practice*. USA: Prentice Hall Press, 6th ed., 2013.

[14] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[15] LogRhythm, "The apt lifecycle and its log trail," *Tech. Rep.*, July 2013.

[16] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions," *The Journal of Supercomputing*, vol. 75, pp. 4543–4574, 2019.

[17] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152, IEEE, 2019.

[18] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 1777–1794, 2019.

[19] W. Z. A. Zakaria, M. F. Abdollah, O. Mohd, and A. F. M. Ariffin, "The rise of ransomware," in *Proceedings of the 2017 International Conference on Software and e-Business*, pp. 66–70, 2017.

[20] D. Strom, "Beware of blackbyte ransomware," *Avast Blog*, February 2022.

[21] H. C. YÜCEEL, "Ttps used by blackbyte ransomware targeting critical infrastructure," *Picus Blog*, February 2022.

[22] T. M. Research, "Revil," *Trend Micro*, 12 2021.

[23] K. Dilanian, "Code in huge ransomware attack written to avoid computers that use russian, says new report," *NBC News*, 07 2021.

[24] C. O. Release, "Blackmatter ransomware," *Cybersecurity & Infrastructure Security Agency*, October 2021.

[25] W. Labs, "Blackmatter ransomware targets food/agriculture sector," *Food Engineering Magazine*, 10 2021.

[26] T. D. REPORT, "Real intrusions by real attackers, the truth behind the intrusion: Diavol ransomware," *THE DFIR REPORT*, 12 2021.

[27] D. Neemani and A. Rubinfeld, "Diavol - a new ransomware used by wizard spider?," *FORTINET Blog*, 07 2021.

[28] L. Abrams, "Fbi links diavol ransomware to the trickbot cybercrime group," *BleepingComputer*, 01 2022.

[29] K. P. Robert McMillan and D. Volz, "Secret world of pro-russia hacking group exposed in leak," *The Wall Street Journal*, 03 2022.

[30] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[31] T. Evgeniou and M. Pontil, "Support vector machines: Theory and applications," vol. 2049, pp. 249–257, 01 2001.

[32] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The journal of educational research*, vol. 96, no. 1, pp. 3–14, 2002.

[33] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[34] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[35] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000277–000282, 2017.

[36] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *ACM SIGKDD explorations newsletter*, vol. 2, no. 2, pp. 81–85, 2000.

[37] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, Ieee, 2009.

[39] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International journal of advanced research in computer and communication engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[40] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," in *International workshop on recent advances in intrusion detection*, pp. 173–191, Springer, 2003.

[41] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining mud policies with sdn for iot intrusion detection," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, pp. 1–7, 2018.

[42] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park, "Kargus: a highly-scalable software-based intrusion detection system," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 317–328, 2012.

[43] M. Alkasassbeh, G. Al-Naymat, A. Hassanat, and M. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 1, pp. 436–445, 2016.

[44] A. Halimaa and K. Sundarakantham, "Machine learning based intrusion detection system," in *2019 3rd International conference on trends in electronics and informatics (ICOEI)*, pp. 916–920, IEEE, 2019.

[45] P. K. Keserwani, M. C. Govil, E. S. Pilli, and P. Govil, "A smart anomaly-based intrusion detection system for the internet of things (iot) network using gwo–pso–rf model," *Journal of Reliable Intelligent Environments*, vol. 7, no. 1, pp. 3–21, 2021.

[46] E. S. P. Krishna and T. Arunkumar, "Hybrid particle swarm and gray wolf optimization algorithm for iot intrusion detection system," *International Journal of Intelligent Engineering & Systems*, 2021.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[48] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[49] A. Rahali and M. A. Akhloufi, "Malbert: Using transformers for cybersecurity and malicious software detection," *arXiv preprint arXiv:2103.03806*, 2021.

[50] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *international symposium on recent advances in intrusion detection*, pp. 382–404, Springer, 2015.

[51] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "Shieldfs: a self-healing, ransomware-aware filesystem," in *Proceedings of the 32nd annual conference on computer security applications*, pp. 336–347, 2016.

[52] T. K. Tran and H. Sato, "Nlp-based approaches for malware classification from api sequences," in *2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*, pp. 101–105, IEEE, 2017.

[53] P. Najafi, D. Koehler, F. Cheng, and C. Meinel, "Nlp-based entity behavior analytics for malware detection," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–5, IEEE, 2021.

[54] Q. Wang, H. Yan, and Z. Han, "Explainable apt attribution for malware using nlp techniques," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pp. 70–80, IEEE, 2021.

[55] P. Security, "Picus Security the complete security control validation platform,"

[56] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," in *Technical report*, The MITRE Corporation, 2018.

[57] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck®: Design and philosophy," 2020.

[58] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.