mathematics

MDPI

Article

Securing IoT Devices Running PureOS from Ransomware Attacks: Leveraging Hybrid Machine Learning Techniques

Tariq Ahamed Ahanger [1,*], Usman Tariq [1,*], Fadl Dahan [2], Shafique A. Chaudhry [3] and Yasir Malik [4]

[1] Management Information System Department, College of Business Administration, Prince Sattam Bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia
[2] Department of Management Information Systems, College of Business Administration-Hawtat Bani Tamim, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia; f.naji@psau.edu.sa
[3] Reh School of Business, Clarkson University, Potsdam, NY 13699, USA; schaudhr@clarkson.edu
[4] Department of Computer Science, Faculty of Science, Bishops University, 2600 Rue College, Sherbrooke, QC J1M 1Z7, Canada; ymalik@ubishops.ca
* Correspondence: t.ahanger@psau.edu.sa (T.A.A.); u.tariq@psau.edu.sa (U.T.); Tel.: +966-(11)-5887080 (T.A.A. & U.T.)

Abstract: Internet-enabled (IoT) devices are typically small, low-powered devices used for sensing and computing that enable remote monitoring and control of various environments through the Internet. Despite their usefulness in achieving a more connected cyber-physical world, these devices are vulnerable to ransomware attacks due to their limited resources and connectivity. To combat these threats, machine learning (ML) can be leveraged to identify and prevent ransomware attacks on IoT devices before they can cause significant damage. In this research paper, we explore the use of ML techniques to enhance ransomware defense in IoT devices running on the PureOS operating system. We have developed a ransomware detection framework using machine learning, which combines the XGBoost and ElasticNet algorithms in a hybrid approach. The design and implementation of our framework are based on the evaluation of various existing machine learning techniques. Our approach was tested using a dataset of real-world ransomware attacks on IoT devices and achieved high accuracy (90%) and low false-positive rates, demonstrating its effectiveness in detecting and preventing ransomware attacks on IoT devices running PureOS.

Keywords: ransomware detection; machine learning; malware analysis; feature extraction; Internet of Things (IoT)

MSC: 68M25

Citation: Ahanger, T.A.; Tariq, U.; Dahan, F.; Chaudhry, S.A.; Malik, Y. Securing IoT Devices Running PureOS from Ransomware Attacks: Leveraging Hybrid Machine Learning Techniques. *Mathematics* **2023**, *11*, 2481. https://doi.org/10.3390/math11112481

Academic Editor: Wei Fang

Received: 17 April 2023
Revised: 20 May 2023
Accepted: 25 May 2023
Published: 28 May 2023

1. Introduction

1.1. Background on Ransomware Attacks

The Internet of Things (IoT) is causing a significant transformation in the way people live and work. The prevalence of internet-connected devices in households is rising, including but not limited to smart thermostats, light bulbs, speakers, and virtual assistants, which can be remotely controlled through mobile devices. IoT devices are used extensively across various industries, e.g., mining, utilities, agriculture, automotive, discrete manufacturing, etc., to collect data at various stages of operations to leverage artificial intelligence (AI) and predictive analytics [1]. Incorporating these sensors enables monitoring and control of a process or environment in real-time, resulting in faster and more rational decision-making.

Although IoT devices have immense potential, their vulnerability to network attacks remains a significant concern. Network threats, such as data theft, phishing attempts, spoofing, and denial of service, can affect IoT devices. These attacks can lead to additional cybersecurity risks, such as ransomware, which can be incredibly expensive and time-consuming to fix for enterprises. The number of ransomware attacks has surged in recent years. One notable incident was the WannaCry ransomware attack in 2017, which affected a large number of

computers globally, including many IoT devices [2]. Another incident in 2019 targeted a smart building in Finland, which caused considerable damage [3]. In 2020, a German hospital was also affected by ransomware that targeted an IoT device, resulting in the shutdown of critical systems, including emergency services. A report recently published by Sonic Wall highlighted a 77% increase in malware attacks on IoT devices during the first half of 2022 [4]. According to the report, ransomware attacks had decreased by 23%, whereas cryptojacking attacks had increased by 30%, and intrusion attempts had increased by 19%. These numbers point to the growing threat of ransomware attacks on IoT devices and underscore the need for more robust security measures to handle such attacks.

1.2. The Need for Effective Defense Mechanisms against Ransomware Attacks

We assert that IoT devices require effective protection measures due to their characteristics as well as their applications. The following are some of the reasons that support our assertion:

(a) Due to their compact and low-cost form factors, many devices in the IoT suffer from processing power and memory constraints. They may not have the resources to run computer-intensive security programs or communicate at a high bandwidth. Therefore, they become increasingly susceptible to ransomware anomalies as the number of linked devices grows.
(b) Because of a lack of robust security measures and standards, many IoT devices are vulnerable to attacks. This is a real concern, especially for older devices that were not always built with safety in mind.
(c) Sensitive information, such as medical records, financial records, and personal preferences, is frequently collected by IoT devices. These sensors' data could be stolen and utilized for nefarious purposes if they were hacked.
(d) The hardware, software, and network architecture that make up an IoT system can be rather complicated. Because of this complexity, proactively spotting and preventing ransomware is challenging. Due to heterogeneous operational and functional requirements, integrating IoT equipment into older, less secure systems is widespread. Therefore, it could be challenging to protect these systems without causing operational disruptions.

Despite all these challenges, putting security first is essential for the IoT devices to realize a secure IoT paradigm.

1.3. The Role of Machine Learning in Ransomware Defense

Machine learning (ML) can play an important role in ransomware defense in IoT by helping to detect and prevent ransomware attacks before they can cause significant damage. For example, ML algorithms can be trained to recognize patterns in IoT network traffic that may indicate that a malware attack is potentially underway. This can include detecting unusual network behavior, such as a sudden surge in traffic or a large number of requests for a particular type of data. ML models can be trained on the existing attacks data and be used to predict/identify similar attacks in the future. Several predictive modeling systems have been developed for malware detection such as:

(a) Random Forest algorithm with an ensemble of decision trees was used to classify malware samples in [4].
(b) Support Vector Machine (SVM) is a supervised learning algorithm that has been used for classification and regression analysis [5,6].
(c) In probability theory, Bayes' theorem is the basis for the Naive Bayes algorithm and has been used in spam detection to identify malware [6].
(d) Decision trees [7] are another ML technique that has been frequently employed in combination with other supportive algorithms for malware detection.
(e) Logistic regression [8] is a statistical method used to figure out how likely a binary outcome is to happen. It has been used successfully in programs that look for malware.
(f) Neural Networks [9] have also been used successfully in malware detection applications.

Traditionally, researchers use various features to train machine learning models to identify the signatures or behaviors of malware. These models are then used to create a framework that could identify and mitigate specific anomalies such as ransomware. The following are some of the widely used factors that are used to train ML models for malware-detecting systems:

(a) Unusual or high-volume network traffic [10], as well as traffic from unknown sources, ports, or protocols, are just some of the indicators that were uncovered by ML models monitoring network activity.
(b) System calls are used by malware to communicate with the operating system and were a telltale sign of malicious software [11]. Models trained with ML were very vigilant on system calls for signs of malicious activity.
(c) Resource use anomalies [12] caused by malware, such as high central processing unit (CPU) or memory usage, were easily detectable by ML models.
(d) Anomalous activity, such as changes to system settings [13] or user behavior that does not make sense, might be a telltale sign of malware and was detected using ML models.
(e) The software on IoT devices was analyzed by ML models for the presence of recognized malware signatures or dangerous patterns.

1.4. PureOS

PureOS is an open-source operating system based on the Linux kernel and includes pre-installed privacy-enhancing tools, such as the Tor Browser and hypertext transfer protocol secure (HTTPS-Everywhere) and has strong default encryption for user data. PureOS has a built-in feature. "PureBoot", that uses a "Heads" firmware payload to enable a user to boot the system from a trusted source and check the integrity of the system's firmware and boot process. PureBoot is a great way to establish an effective measure for preventing malware installation on a device.

Like any operating system, PureOS is also a target of "unpatched security flaws", "misconfigured settings", "weak authentication", "social engineering vulnerabilities (e.g., fake software updates, etc.)", and "supply-chain attack (e.g., inserting backdoors or other malicious code during the manufacturing or distribution process)". Ultimately, any successful anomalous attempt can trigger an enterprise-wide impact that may reflect the horrific consequences of ransomware.

The main objective of this paper is to put forward and investigate solutions to mitigate the impact of ransomware vulnerabilities on IoT devices that run PureOS [14]. The following are the main contributions of this work:

i. We investigated 15,000 samples (i.e., ransomware and benign) instances, detailing hitherto unreported facets of ransomware attacks with an emphasis on shared traits amongst malware families.
iii. We outlined the design process behind the fundamental components of ransomware samples and discussed how this knowledge can be leveraged to prevent future intrusion. In devastating ransomware cyberattacks of varying degrees of complexity, our research demonstrated that aberrant control efforts should be reliably monitored.
iii. We proposed methods to counter the widespread threat of dissimilar ransomware attacks. We have suggested a generic approach to detecting such risks, one that makes no presumptions about the specific methods through which user records are maliciously made unavailable.

The rest of the paper is organized as follows. Section 2 is dedicated to presenting a comprehensive literature review, while Section 3 delves into the intricate details of data collection, augmentation, balancing, and processing techniques. In Section 4, we present our approach, while Section 5 expounds upon the practical implementation and rigorous testing of our proposed ransomware analysis and identification architecture. Ultimately, the paper culminates in Section 6, where a conclusion is reached.

2. Literature Review

The NIST 2018 framework [15] proposes the adoption of a Framework Core consisting of five fundamental functions, i.e., Identify, Protect, Detect, Respond, and Recover, to structure cybersecurity activities optimally. These elements aid organizations in formulating their cybersecurity risk management strategy by arranging data, supporting risk management decisions, reducing risks, and enhancing performance through the integration of previous experiences.

Organizations are mandated by the NIST guidelines to implement targeted strategies to combat malware effectively. These strategies encompass various aspects, including the timely identification and characterization of incidents, the swift dissemination of pertinent information, evaluation of actions that may hinder recovery efforts, reinforcement of information sharing within network environments, implementation of corrective measures to prevent a recurrence, monitoring of precursor events or indicators for future incident detection, and the acquisition of supplementary tools and resources for incident detection, analysis, and mitigation. By proactively adopting these measures, organizations can fortify their systems against potential threats and maintain their resilience in the face of cyber-attacks. From the earliest extensive analyses of ransomware behavior [16,17], scholars have advanced diverse perspectives and multifarious tools and techniques to detect ransomware behavior, including but not limited to filesystem activity monitoring and application programming interface (API) hooking. It is significant to note that while static analysis, particularly signature-based detection, retains its status as a conventional method for detecting malware in general, it is not as widely utilized in the context of ransomware detection. Despite many antivirus tools incorporating ransomware signatures into their databases, current research primarily accentuates the significance of behavioral approaches, potentially in response to the ubiquitous adoption of ransomware-as-a-service (RaaS) and the inclination of ransomware authors to imitate one another, resulting in the emergence of a profusion of dissimilar and transient variations.

The increasing prevalence of ransomware among attackers has led to a surge in its popularity within the realm of cybersecurity research. Upadhyaya et al. [18] conducted a comprehensive analysis of the anatomy and features of ransomware, a type of malicious software that frequently blocks access to task manager, command prompts, and other executable files, rendering the infected system unusable. Nevertheless, the present study focuses exclusively on CTB Locker, a specific type of ransomware, and explores its modus operandi in terms of infiltration, its process of generating a Bitcoin wallet for each target, and its payment system facilitated through the Tor network. Meanwhile, certain physicists have suggested the implementation of quantum cryptography systems that are impervious to loopholes, which have been compared to illusory mirages. Conversely, others advocate taking proactive measures such as safeguarding digital assets and maintaining routine backups in preparation for any future attacks. In Gagneja's [19] analysis, several methods are identified by which ransomware infiltrates a system by exploiting security vulnerabilities within outdated applications on a victim's computer. As a consequence of such an attack, backup files and directories are deliberately removed to obstruct the system's restoration process, leading to the eventual encryption of vital system files. To counteract these malicious activities, it was recommended to provide comprehensive training to personnel on all matters related to system security, ensuring the timely installation of patches to address any potential security weaknesses, implementing firewall protection, conducting regular email scanning, and employing only licensed operating systems as preventative measures against the possibility of ransomware attacks.

Celdrán and Moon, in their respective works [20,21], present an evaluation of the impact of various techniques such as hash-coded string extraction, file format analysis, file fingerprinting, packer detection, and disassembly on the efficacy of static and dynamic analysis. The primary objective of this analysis was to yield two critical advantages. The first advantage is the safety that static analysis affords during the evaluation process, given that there is no need to execute the malware. Secondly, the method provides more

profound insights into the execution pathways of malware, enabling a more comprehensive understanding of its operations. Furthermore, the research illustrated that in the realm of binary analysis, two primary methodologies can be employed for malware analysis: static analysis and dynamic analysis. Static analysis involves scrutinizing the binary without execution as a preliminary step. This approach does not necessarily necessitate the utilization of a virtual environment and can be challenging to utilize with packed binaries unless they are unpacked manually. However, static analysis is capable of rendering an extensive and all-encompassing view of the code coverage with a low false positive rate. Conversely, dynamic analysis requires the binary to be executed first before being analyzed. To start the analysis process, a virtual environment must be configured, and packed binaries are automatically unpacked. While dynamic analysis provides insight into the path of execution of running modules, its false positive rate is notably high.

Dargahi et al. [22] formulated a systematic classification of the distinguishing attributes of ransomware from the perspective of cybercriminals using the Cyber Kill Chain (CKC) model. This work explores the interconnectedness between various ransomware characteristics and the different stages of the Cyber Kill Chain (CKC). It focuses on how factors such as payload delivery and access prevention play a role throughout the CKC, starting from the weaponization phase and progressing until the desired objectives are achieved. Although Dargahi et al.'s approach is innovative, its scope was narrow. The authors solely analyzed crypto-ransomware that targets desktop systems and its malevolent attributes, such as the potentiality of botnet deployments. The authors did not assess the efficacy or feasibility of alternative strategies nor explore mobile or IoT platforms, which can be susceptible to ransomware attacks.

Furthermore, it is essential to note that the taxonomy proposed by Dargahi et al. [22] is only one of several approaches to categorizing ransomware. Other researchers have proposed alternative taxonomies that focus on different aspects of ransomware behavior, such as the analysis of network traffic or the identification of ransomware families based on code similarities. While Dargahi et al.'s method was valuable in identifying the objectives and motives of ransomware attackers, it did not provide insights into the best practices for preventing or mitigating ransomware attacks.

Table 1 provides a comprehensive overview of different ransomware detection techniques, presented in existing works, and their respective features, advantages, and disadvantages. Signature-based techniques are well-established and effective against known ransomware variants, but can be ineffective against new or polymorphic variants. Heuristic-based techniques can detect new or unknown variants, but may have a higher false-negative rate and limited ability to differentiate between benign and malicious activity. Machine-learning-based techniques offer the ability to learn and adapt to new variants, but require significant amounts of representative data and may produce false positives. Hybrid approaches provide a combination of signature and machine-learning-based techniques for improved accuracy, but can be resource-intensive. It's important to note that the effectiveness of each technique may vary depending on the specific implementation, the ransomware being targeted, and the context in which the detection is taking place.

Overall, we have reviewed 298 research papers that were searched with the keyword "Ransomware" on Google Scholar that were published from the year 2010 to April 2023, and we have found a few issues that have not been properly covered in existing research. The first issue we encountered pertains to the widespread and interchangeable usage of the term "ransomware" and "crypto-ransomware", which may indicate a lack of consensus among researchers as to whether these two terms are technically equivalent or whether non-crypto-ransomware can be classified as ransomware at all.

Another issue we identified is the lack of a universal standard for defining benign or malicious (ransomware-like) behaviors. Ransomware is a type of malware that is primarily designed to extort ransom payments from users, and while it is generally agreed that different variants of ransomware share two common features, namely, blocking user access to resources (often files) and attempting to extort ransom payments, researchers have

divergent views on which additional features or feature combinations are indicative of malicious behavior.

Table 1. Comparison of ransomware detection techniques and their features, advantages, and isadvantages.

Detection Technique	Features	Advantages	Disadvantages
Signature-based	Hash values, file names, behavior patterns	High accuracy, low false positive rate	Inability to detect new, unknown ransomware variants, ineffective against polymorphic ransomware
Heuristic-based	Behavior patterns, file access patterns, network traffic	Ability to detect new, unknown ransomware variants, low false positive rate, effective against polymorphic ransomware	Higher false negative rate, limited ability to differentiate between benign and malicious activity
Machine learning-based	Dynamic behavior analysis, system calls, network traffic, entropy, header information	Ability to detect new, unknown ransomware variants, ability to differentiate between benign and malicious activity, high accuracy, effective against polymorphic ransomware	Requires large, representative datasets for training, may be susceptible to adversarial attacks, may produce false positives due to benign software with similar behavior
Hybrid approach	Combination of signature-based and machine-learning-based techniques	Improved accuracy and ability to detect new, unknown ransomware variants, effective against polymorphic ransomware	May be more complex and resource-intensive, may still miss new, unknown ransomware variants

Furthermore, we observed a lack of uniform usage of terminologies in the context of mitigation strategies, which could potentially lead to confusion and misunderstandings. Finally, we found that there is no universally accepted standard for evaluating and comparing the effectiveness of different strategies, further underscoring the need for additional research in this area.

3. Data Collection and Preparation

3.1. Data Collection and Processing Techniques

In this section, we discuss our procedure for selecting ransomware samples, which played a vital role in our study's malware dataset collection. To assemble the ransomware datasets, we utilized the widely used malware analyzer Anubis and ESET NOD32 [23], as well as a plethora of publicly available malware archives and anecdotal research in online security forums. We compiled and analyzed a dataset comprising more than 15,000 ransomware samples. Our analysis aimed to uncover novel insights into previously undocumented aspects of ransomware attacks and identify commonalities among different malware families. To ensure the validity and precision of the dataset, we conducted a rigorous examination of multiple factors. These included assessing the reliability and diversity of the data sources, evaluating the size and diversity of the dataset, verifying the accuracy of pre-labeled ransomware classifications, performing meticulous data preprocessing and normalization, ensuring the integrity of the data, and considering the timeframe during which the data were collected. The validation process involved meticulous checks for inconsistencies, errors, and biases within the dataset. Furthermore, we compared our dataset with other publicly available datasets or ground truth data to further validate its reliability.

We have adopted the dynamic analysis technique that involved running ransomware in a controlled environment such as a sandbox and virtual machine to observe its behavior and capture relevant data such as system calls, network traffic, and registry modifications. The dynamic analysis helped us understand the dissimilar characteristics of the ransomware dataset. Ransomware characteristics include, but are not limited to, metadata, behavior

logs, network traffic, malware landscape, representativeness (i.e., ransomware families, types, and variants), transferability of threat models, imbalance (i.e., data noise and errors), and temporality (i.e., time-period and frequency of malware sample collection).

3.2. Preprocessing and Feature Engineering

For data preprocessing and feature engineering of ransomware, we have removed duplicate and irrelevant data, handled missing values, and scaled the ransomware log data to ensure that features are comparable. Effective data preprocessing and feature engineering improve the accuracy of ransomware detection and facilitate the development of a robust security framework. Feature engineering involved the examination of file size, file type, file entropy, API calls, code obfuscation, code analysis, sandbox analysis, and evaluation of digital signatures if files are digitally signed by the creator for feature detection of malware.

3.3. Data Augmentation and Balancing Techniques

For ransomware data augmentation, we applied two techniques: (a) random noise (i.e., adding random noise (e.g., irrelevant features, redundant features, missing codes, and significantly different data points) to the malware samples to make them more robust to variations in the data), and (b) random cropping (i.e., cropping the malware samples to a smaller size). The purpose of data augmentation for ransomware was to increase the size and variability of the malware dataset used to train the applied machine learning model. By generating new samples from the existing dataset through data augmentation techniques, we were able to create a more diverse and representative training set, which led to the proposed method to improve model performance and generalization.

Consequently, data augmentation helped to address the problem of imbalanced datasets, where the number of samples in each class was not equal. We employed the Synthetic Minority Over-sampling Technique (SMOTE) to balance the dataset, by generating synthetic samples that increased the number of minority class samples. This helped us tackle the problem of class imbalance, where there were significantly fewer samples in the minority class compared to the benign class. SMOTE also prevented overfitting, as it increased the size of the minority class, leading to a better generalization of the model to new data. Consequently, SMOTE was instrumental in accurately classifying new malware samples.

Table 2 exhibits the pseudocode that outlines how Synthetic Minority Over-sampling Technique (SMOTE) can be applied to identify ransomware in a dataset.

Table 2. Assessing the feasibility of SMOTE for ransomware detection in a dataset.

SMOTE Applicability to Identify Ransomware in a Dataset
(1) Load the ransomware dataset.
(2) Split the dataset into training and testing sets.
(3) Determine the minority class (i.e., ransomware samples).
(4) Apply SMOTE to the training set to generate synthetic samples for the minority class:
(a) Determine the number of synthetic samples to generate based on the desired ratio of minority to majority samples.
(b) Select a random minority sample.
(c) Identify its *k* nearest neighbors.
(d) Randomly select one of the *k* nearest neighbors and use it to create a new synthetic sample by interpolating between the selected sample and its neighbor.
(e) Repeat steps b–d until the desired number of synthetic samples is generated.
(5) Combine the original training set with the generated synthetic samples to create a new, balanced training set.
(6) Train a machine learning model on the new training set.
(7) Evaluate the model's performance on the testing set, using metrics such as precision, recall, and F1-score.
If the model's performance is satisfactory, use it to predict whether new samples are ransomware or not.

3.4. Focused Ransomware Variants

To evaluate the effectiveness of our proposed ransomware detection framework in a real-world scenario, we needed to test it on actual ransomware samples. However, the ransomware variants we had access to were not compatible with the PureOS operating environment. Therefore, we re-implemented the ransomware variants to make them suitable for the PureOS environment. This process involved analyzing the following ransomware code and modifying it to ensure that it could be executed and studied within the PureOS environment.

i. The **Kryptik** [24] ransomware is a type of malware that is often disseminated through email phishing campaigns and exploit kits. This advanced form of ransomware uses encryption algorithms to lock down the victim's files, rendering them inaccessible. Kryptik ransomware was re-designed to evade detection by antivirus software (i.e., Virus Chaser [25]) and uses command-and-control (C&C) servers to obtain instructions from the attacker. It employs encryption algorithms (i.e., RSA-2048 and AES-256) to encrypt the victim's files, rendering them inaccessible. It utilizes obfuscation techniques to conceal its activities. The impact of Kryptik ransomware can be catastrophic, resulting in critical data loss and disrupting business operations.

ii. We have re-implemented the **Cloud Snooper** [26] ransomware to target cloud-based systems and services (i.e., Tonido cloud platform [27] through the Nautilus file manager plugin). It exploited the weaknesses in cloud infrastructure to gain unauthorized access to the victim's network. Some of the notable features of Cloud Snooper ransomware include its ability to bypass firewalls and intrusion detection systems and encrypt files. It operated covertly to evade detection and caused severe damage to the victim (i.e., sandbox experimental setup. The impact of Cloud Snooper ransomware was particularly devastating, as it resulted in the loss of sensitive information and disruption of normal OS operations (i.e., encrypting or locking files, modifying system settings, and interfering with the normal functioning of applications and system processes).

iii. The **WannaCry** [28] ransomware was first identified in May 2017. It spread rapidly, infecting over 230,000 computers in over 150 countries within just a few days. Originally, the ransomware used a vulnerability in Microsoft Windows known as EternalBlue to spread from one computer to another, making it particularly dangerous. Key features of WannaCry were as follows:

    a. It encrypts files on the infected system using the AES encryption algorithm, making them inaccessible to the user.
    b. It can spread rapidly across a network, infecting other vulnerable computers without any user interaction.
    c. A "kill switch" was built into the code of WannaCry, allowing researchers to halt the spread of the ransomware by registering a domain name that the malware checked before encrypting files.

    WannaCry was altered and reprogrammed to accommodate the PureOS functional requirements that were originally implemented to specifically targeted systems running Microsoft Windows operating systems, with a particular focus on older, unsupported versions such as Windows Server 2003 and Server 2022. The ransomware payload was delivered as a PureOS executable file disguised as a software update. Once the file was executed, it installed ransomware on the system and began encrypting files. We have used AES encryption to encrypt files on the infected system, with a unique key generated for each system. The re-implemented ransomware also encrypted the key itself using RSA encryption, making it intolerable to decrypt the files without the private key presumably held by the attackers.

iv. **LockBit** [29] is a file-encrypting ransomware that uses a combination of RSA and AES encryption algorithms to encrypt the victim's files. Once the files are encrypted, the ransomware displays a ransom note, demanding payment in exchange for the decryption key. We re-designed the malware by granting it the ability to spread across

a network and infect multiple devices connected to it. Revised implementation was equipped with a timer feature that deletes files after a set amount of time, which means that the anomaly must be counter-measured to diminish the impact. This ransomware was keen to target critical files, such as documents, images, and databases.

v. Re-programmed Black Basta [30] ransomware used AES-256 encryption to encrypt files on the victim's PureOS mounted computer (i.e., including desktops, laptops, and servers). It appended a unique extension to encrypted files, making them unusable until they are decrypted. The encryption process took several minutes or in some iterations even hours, depending on the size of the files.

vi. **Revised Hive** [31] ransomware used a combination of RSA and AES encryption algorithms to lock the victim's files (i.e., experimental setup). It entered the system through an exploit kit and could spread to other connected devices on the sandbox network. The ransomware could erase shadow copies and backup files to obstruct the victim's efforts to recover their encrypted data.

vii. **ALPHV**, **BlackCat**, and **Noberus** [31] are three distinct ransomware families with their own unique features, system and network targets, technical details, and impact. Common features included its use of double extortion tactics, which involve not only encrypting a victim's files (i.e., AES-256, and RSA), but also stealing sensitive data. We re-implemented these ransomware variants by using multiple techniques to evade detection, including code obfuscation, anti-debugging techniques, and process injection. During certain experimental iterations, we appended the ".noberus" extension to encrypted files. We have observed that ransomware typically appends a unique extension to encrypted files as a way to differentiate them from their original unencrypted state.

viii. PureOS-focused **AvosLocker** [32] used strong encryption algorithms (such as AES-256 and RSA) to encrypt files on a victim's computer or network. AvosLocker targeted the honeypot computer and network that was vulnerable to its distribution method (such as outdated Remote Desktop Protocol (RDP)) and contains vulnerabilities that can be exploited. In the revised implementation, AvosLocker generated a unique encryption key for each infected computer, which was stored on the attacker's (i.e., anomaly) server. The impact of this ransomware was severe, as it caused the victim to lose access to important files and data.

ix. The **Conti** [33] ransomware is a highly advanced and complex malware that uses a sophisticated encryption algorithm to encrypt files on a victim's computer system. It can spread through a network, infecting other connected systems. The vulnerabilities that Conti exploits in PureOS include exploiting weaknesses in the RDP protocol to gain access to internet-connected systems, exploiting vulnerabilities in VPN and remote access software such as Pulse Secure VPN, Fortinet VPN, and Citrix ADC, and exploiting vulnerabilities in web servers such as Apache and Nginx to gain unauthorized access to victims' systems. To achieve our goal, we have ensured that Conti ransomware uses a combination of symmetric and asymmetric encryption techniques to encrypt the files of its victims (i.e., random 256-bit ChaCha symmetric key for each file's encryption and an asymmetric encryption algorithm RSA cryptography for the encryption of the ChaCha key). Furthermore, it communicated with its C&C server using an encrypted channel, making it difficult to track its activities.

x. We implemented **REvil** [34] ransomware more powerfully by using stronger encryption algorithms such as RSA-2048 and AES-256. This allowed the ransomware to encrypt not only local files but also files on network shares and mapped drives. As a result, any PureOS-based computing system, including the Librem Server, workstations (such as the Librem 14 and Librem Mini), and cellular devices (such as the Librem 5) could potentially be targeted [35]. After infecting a victim's computer, the ransomware was designed to remain there by creating a scheduled task or modifying the registry. Furthermore, we made the ransomware even more malicious by adding the ability to exfiltrate sensitive data before encrypting it.

xi. We implemented **DarkSide** [35] ransomware by enforcing strong encryption algorithms, such as RSA and AES, to encrypt files on a victim's computer and prevent them from being accessed without meeting the adversary criteria. Various obfuscation techniques (i.e., (a) code encryption and obfuscation, (b) applying the polymorphic code, (c) applying the dynamically linked to system libraries, (d) malware code compression, and (e) equipping it with an anti-debugging capacity to detect when it is being analyzed or debugged and takes actions to evade or disable the analysis) were introduced to evade detection.

xii. The **Babuk** [36] ransomware was re-designed to use a combination of symmetric and asymmetric encryption algorithms to encrypt data on the target system. It used a per-file random 256-bit ChaCha symmetric key for each file's encryption, and an asymmetric encryption algorithm such as RSA cryptography for the encryption of the ChaCha key. The asymmetric encryption algorithm is used to securely transmit the ChaCha key to the ransomware operator, allowing them to decrypt the files. Babuk's feature allowed it to steal data from infected systems. These data were then encrypted and sent to the ransomware operator (i.e., adversarial process). It was also capable of terminating running processes, deleting shadow volume copies, and disabling the PureOS System Restore feature.

xiii. To satisfy the experimental requirement, we redesigned the **Egregor** [37] ransomware that enabled it to use a mix of symmetric and asymmetric encryption algorithms to encrypt files on the targeted computer. The process involved generating a unique 256-bit ChaCha symmetric key for each file and using the RSA algorithm to encrypt the ChaCha key for secure transmission to the attacker (i.e., automated process), which could then decrypt the files. Moreover, the ransomware had various capabilities such as appending a random extension to the encrypted files, exploiting vulnerabilities in RDP connections and exploit kits, stealing data from infected computers, terminating processes, removing shadow volume copies, and disabling the PureOS System Restore function. Upon infecting the computer, the ransomware compressed the encrypted files into a single archive using an encryption and compression technique (i.e., "Lossless" and "Huffman coding" [35] compression).

xiv. The updated/re-designed version of the **Avaddon** [37] ransomware had numerous functions, such as employing both symmetric and asymmetric encryption methods to encrypt files. It used the RSA algorithm to encrypt files and used an exclusive AES-256 key for each file, making it challenging to decrypt without the key. The ransomware also added a distinct extension to each encrypted file, making it hard to recognize and retrieve the files. Moreover, the malware was equipped with an extended capacity to extract sensitive data from the infected system and forward it to the attacker node (i.e., an automated process). It could terminate ongoing processes and deactivate various operating system security features, including PureBoot [38].

4. Applied Machine Learning Models for Ransomware Defense

4.1. Overview of Different Machine Learning Models

Detecting malware using machine learning is a complex undertaking that has been a focus of research for many years. With the increasing sophistication of ransomware, there is a constant race between security researchers and malware creators to stay ahead of each other. This means that research in this field will always remain important and relevant. Even if a new machine learning method is developed that is capable of identifying all types of ransomwares, it is likely that malware creators will eventually develop new techniques to evade detection. As a result, the pursuit of improving malware detection methods is an ongoing process that requires continuous innovation and adaptation to keep up with the evolving threat landscape.

Our study aimed to identify the most effective ML approach(es) for detecting ransomware and benign executable files. To achieve this, we have adopted a three-step methodology.

Firstly, we conducted a thorough review of state-of-the-art machine learning methods (random forest, support vector machine (SVM), decision tree, naïve Bayes, AdaBoost, etc.) and examined the datasets and data collection methods used in recent research to identify the most promising techniques.

Secondly, we re-implemented and trained the three most effective methods identified in the first step using our collected dataset. By re-implementing and training these methods, we aimed to assess their performance on our specific dataset and compare their accuracy in detecting malicious software.

Ultimately, by using real-world samples, we evaluated the effectiveness of each method in identifying malware and determining which approach offers the best performance for detecting malicious software. Overall, our research endeavored to contribute to the development of a more accurate and reliable ransomware detection method that can enhance cybersecurity and protect against evolving threats.

To accomplish our goal of achieving the desired impact, we performed a comparative analysis of a hybrid-supervised learning approach in three different scenarios. These scenarios were designed to represent different levels of stringency when it came to the samples considered.

(a) The first scenario was very strict, and only a very well-characterized set of samples were included.
(b) The second scenario was less strict and included a broader range of well-studied samples.
(c) The third scenario was the most realistic, representing the actual conditions faced by vendors of ransomware detection solutions.

By designing these three scenarios, we gained insight into how the use of a smaller, more distinct dataset compared to a larger, more varied one can impact the proposed framework. This analysis helped us understand the effects of the framework in a more nuanced way, leading to a better accomplishment of our goal. Overall, our comparative analysis and experimental outcome provided valuable information that helped us make more informed decisions when it comes to implementing supervised learning approaches in real-world scenarios.

4.2. Selection of Appropriate Machine Learning Models

(a) We wanted to find the best machine-learning model for detecting ransomware. Therefore, we created a set of criteria for our search. The criteria are not exhaustive but include the following: the selected model should have high accuracy in detecting ransomware and be able to minimize false positives and false negatives.
(b) The model should be scalable and perform well even when dealing with small or large datasets.
(c) It should be able to generalize well to new and unseen ransomware samples.
(d) The model should be robust and able to perform well in the presence of noise, adversarial attacks, and other anomalies.
(e) The model should provide clear and interpretable explanations for its decisions and predictions.
(f) It should be efficient in terms of computation time, memory usage, and power consumption.
(g) The model should be flexible and easily adaptable to changing ransomware attack patterns with the ability to incorporate new data.

We have explored the relevance of using regression models for detecting ransomware, as they possess the capability to estimate the likelihood of a file or behavior being malicious. This is important, especially because conventional techniques such as signature-based detection may not be effective in detecting new or unknown malware variants. The regression model (XGBoost (i.e., useful for dealing with large datasets and is known for its speed and scalability) [39], and ElasticNet (i.e., to achieve a balance between sparsity and accuracy)) [40] was trained on a dataset of labeled instances, where each example was a file or behavior that was either "malicious" or "benign". By analyzing the features

of these instances, the model could then predict the probability of a new file or behavior being malicious. Some of the initial features that were leveraged for ransomware detection included file size and entropy, the presence of specific strings or signatures, API calls and their arguments, and network traffic patterns. It is worth noting that both XGbBoost and ElasticNet regression have been proven to be useful in machine learning applications where the input data have many features and some of them are correlated. By identifying the key features that are essential for predicting the target outcome, the process of feature selection enhances the practicality of the application.

Referring to Figure 1, in the pseudocode provided above (Table 3), the functions **collect_data()** and **preprocess_data(data)** serve the purposes of data collection and data preprocessing, respectively. The function **split_data(preprocessed_data, test_size = 0.2)** splits the preprocessed data into training and testing sets, **perform_elasticnet(train_data)** identifies the most important features for predicting ransomware using ElasticNet, and **select_features(data, important_features)** selects only the important features from the data. Similarly, the function **train_xgboost(train_data_selected)** trains the XGBoost model on the selected features and **validate_model(model, test_data_selected)** validates the model's performance on the testing data. We used **tune_hyperparameters(model, train_data_selected)** to fine-tune the model's hyperparameters and **evaluate_model_performance(tuned_model_performance)** to obtain the performance evaluation of the tuned model. Finally, **deploy_model(tuned_model)** is used to deploy the tuned model for use in a production environment.

Table 3. Detecting ransomware using XGBoost and ElasticNet.

Pseudocode for Detecting Ransomware Using XGBoost and ElasticNet
(1) Collect and preprocess the data:
(a) data = collect_data()
(b) preprocessed_data = preprocess_data(data)
(2) Split the data:
(a) train_data, test_data = split_data(preprocessed_data, test_size = 0.2)
(3) Feature selection:
(a) important_features = perform_elasticnet(train_data)
(b) train_data_selected = select_features(train_data, important_features)
(c) test_data_selected = select_features(test_data, important_features)
(4) Train the model:
(a) model = train_hybrid_xgboost_elasticnet(train_data_selected)
(b) model_performance = validate_model(model, test_data_selected)
(5) Tune the model:
(a) tuned_model = tune_hyperparameters(model, train_data_selected)
(b) tuned_model_performance = validate_model(tuned_model, test_data_selected)
(6) Evaluate the model:
(a) evaluate_model_performance(tuned_model_performance)
(7) Deploy the model:
(a) deploy_model(tuned_model)

Thus, detecting ransomware using XGBoost involved training a machine learning model using features that helped to distinguish between normal and ransomware behavior. Data related to "file access patterns" was formulated as:

$$XGBoost(Ransomware) = w_1 * num\_of\_files\_created + w_2 * num\_of\_files\_deleted + w_3 * num\_of\_files\_renamed + w_4 * num\_of\_files\_read + b \quad (1)$$

where w_1, w_2, w_3, and w_4 are the weights assigned to the number of files created, deleted, renamed, and read, respectively, and "b" is the bias term.

Data related to "network traffic patterns" was formulated as:

$$XGBoost(Ransomware) = w_1 * num\_of\_outgoing\_connections + w_2 * num\_of\_incoming\_connections + w_3 * num\_of\_data\_packets\_sent + w_4 * num\_of\_data\_packets\_received + b \quad (2)$$

where w_1, w_2, w_3, and w_4 are the weights assigned to the number of outgoing connections, incoming connections, data packets sent, and data packets received, respectively, and b is the bias term. Data related to "system call patterns" was formulated as:

$$XGBoost(Ransomware) = w_1 * num\_of\_system\_calls + w_2 * num\_of\_suspicious\_system\_calls + b \quad (3)$$

where w_1 and w_2 are the weights assigned to the total number of system calls and suspicious system calls, respectively, and b is the bias term.

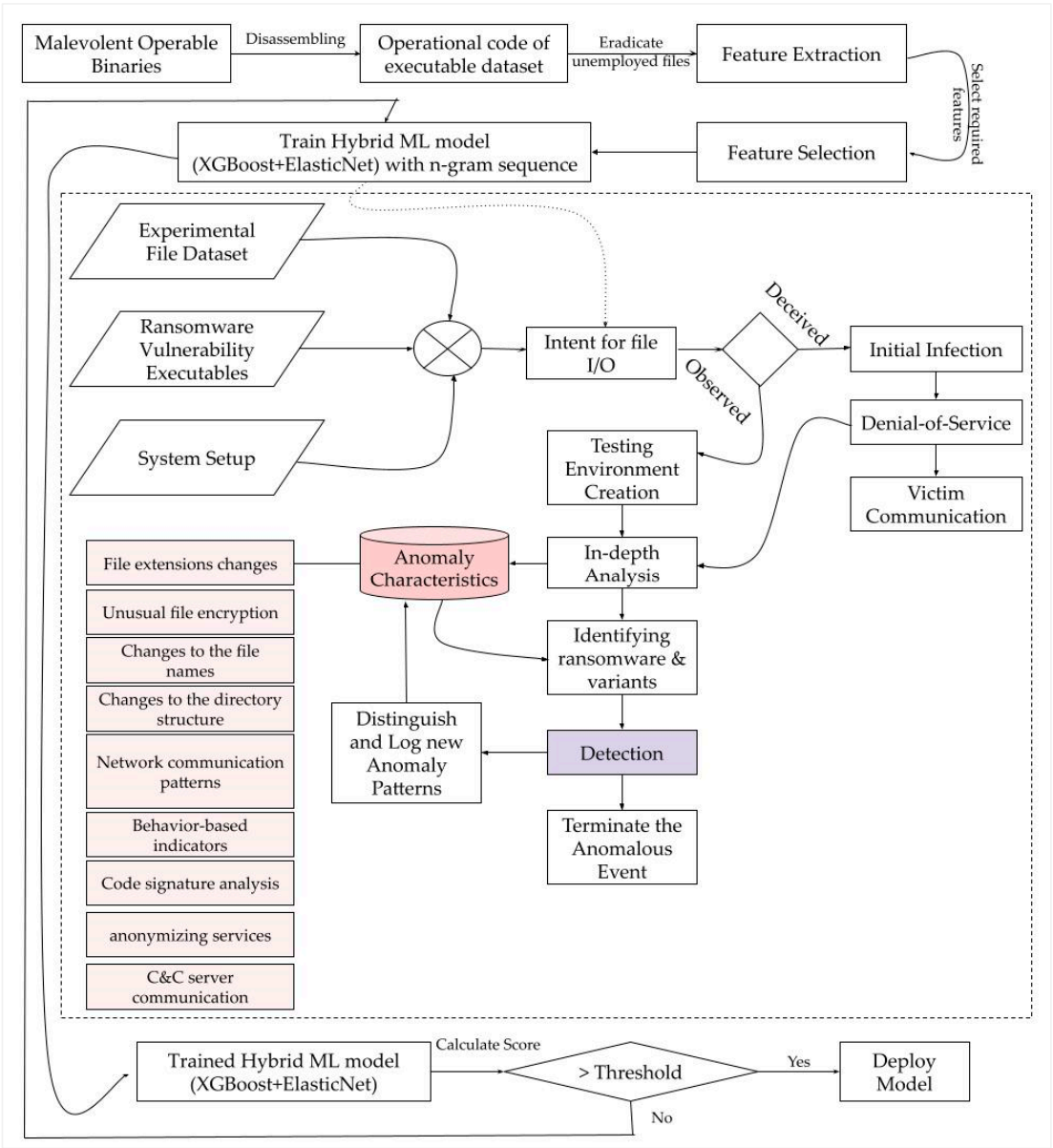

Figure 1. Flow diagram of ransomware detection criteria using XGBoost and ElasticNet.

The experiment was revised with a unique dataset using ElasticNet methodology by following conventional steps (i.e., collection of an indicative dataset, processing the data for feature normalization, splitting data into training and testing sets, training the ML model, and testing the model using a metric such as accuracy, precision, or recall.

ElasticNet loss function was evaluated as:

$$\min\left(1/\left(2 * n_{samples}\right)||y - Xw||_2^2 + alpha * l4_{ratio} * ||w| + 0.5 * alpha * (1 - l4_{ratio}) * ||w||_2^2\right) \quad (4)$$

where

i. "*n_samples*" is the number of samples in the dataset.
ii. "*y*" is the target variable in the dataset.
iii. "*X*" is the matrix of features in the dataset.
iv. "*W*" is the vector of coefficients that are learned by the model.
v. "*14_ratio*" is a hyperparameter that controls the balance between purportedly L_1 and L_2 regularization. As asserted, the L_1 regularization promotes sparsity in the learned coefficients, while L_2 regularization promotes small, non-zero coefficients.
vi. "*Alpha*" is a hyperparameter that controls the strength of the regularization. Higher values of alpha lead to more regularization.

In this scenario, the ElasticNet loss function is referred to as a combination of L_1 and L_2 regularization. The L_1 regularization term is given by $alpha * l4_{ratio} * ||w||_1$, which is the sum of the absolute values of the coefficients multiplied by a scaling factor alpha $*$ l4_ratio. The L_2 regularization term is given by $0.5 * alpha * (1 - l4_{ratio}) * ||w||_2^2$, which is the sum of the squares of the coefficients multiplied by a scaling factor $0.5 * alpha * (1 - l4_{ratio})$. The L_1 scaling factors are designed to balance the strength of the two regularization terms.

4.3. Feature Selection and Model Tuning

To develop effective ransomware detection methods, it was necessary to extract relevant features from the ransomware. This process involved closely analyzing the ransomware's code and behavior to identify specific characteristics or patterns that can distinguish it from other types of malwares. Reimplemented ransomware possesses distinct features that are useful in identifying and detecting them. These features are unique to ransomware and can differentiate it from other types of malwares. Some of the essential ransomware features include, but are not limited to:

i. Atypical network activity that is not typical for the system.
ii. Alterations to file extensions are not typical for the system.
iii. Suspicious processes with names that are random or located in unusual directories.
iv. Changes to the registry.
v. Unusual CPU or disk usage that is not typical for the system.
vi. Pop-up messages or warnings.
vii. Atypical system crashes or errors.
viii. Encryption key generation by malware.
ix. Usage of non-standard encryption algorithms.
x. Unusual behavior, such as modification of file timestamps or the creation of decoy files to deceive the victim.
xi. Atypical file access patterns that are not typical for the system.
xii. Large numbers of file deletions.
xiii. Changes to file permissions that are not typical for the system.
xiv. Random file names on large file datasets, all at once.
xv. Large numbers of failed login attempts.
xvi. Unusual file sizes that are not typical for the system.

Once the relevant features were extracted, they were used to train the machine learning model for detecting and classifying ransomware in real-world scenarios. This approach allowed for more effective ransomware detection, as it leverages the unique characteristics of ransomware to identify and mitigate threats. Furthermore, by continually updating

the feature extraction process, the detection model was adapted to the evolving threat landscape of ransomware attacks.

4.4. Evaluation of Model Performance

To evaluate the performance of our model for detecting ransomware using XGBoost and ElasticNet, we used well-known ML model evaluation metrics, i.e., accuracy, precision, recall, and F1-score. The followings are the main steps performed for the performance evaluation:

1. Split the data into training and testing sets.
2. Perform feature selection using ElasticNet to identify the most important features for predicting ransomware.
3. Train an XGBoost model on the selected features using the training set.
4. Predict the labels of the test set using the trained model.
5. Evaluate the performance of the model.

Table 4 depicts the model code in which **X** and **y are** the features and labels of the data, respectively. **train_test_split()** was used to split the data into training and testing sets. **perform_elasticnet()** identifies the important features using ElasticNet. **select_features()** selects the important features from the data. **train_xgboost()** trains the XGBoost model on the selected features. **predict()** predicts the labels of the test set. Ultimately, the evaluation metrics are computed using the appropriate functions from scikit-learn (**accuracy_score()**, **precision_score()**, **recall_score()**. It is worth highlighting that the importance of employing parameters test_size and random_state is as follows:

i. The test_size parameter specifies the proportion of the data that will be used for testing, while the remaining data are used for training. For example, a test_size of 0.2 means that 20% of the data will be used for testing, and 80% will be used for training.
ii. The random_state parameter is used to set the seed for the random number generator, which ensures that the results are reproducible. This is important because the random sampling of data for training and testing can affect the performance metrics of the model. By setting the random_state parameter to a specific value, the same random sampling will occur every time the code is run, ensuring that the results are consistent and reproducible.

Table 4. Algorithmic outline for assessing model performance.

Pseudocode for Evaluating the Performance of the Model
Split the data into training and testing sets X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42) # *Perform feature selection using ElasticNet* important_features = perform_elasticnet(X_train, y_train) # *Select the important features* X_train_selected = select_features(X_train, important_features) X_test_selected = select_features(X_test, important_features) # *Train an XGBoost model on the selected features* model = train_xgboost(X_train_selected, y_train) # *Predict the labels of the test set* y_pred = model.predict(X_test_selected) # *Compute the evaluation metrics* accuracy = accuracy_score(y_test, y_pred) precision = precision_score(y_test, y_pred) recall = recall_score(y_test, y_pred) f1 = f1_score(y_test, y_pred)

To compute the accuracy, precision, recall, and F1-score for detecting ransomware using XGBoost and ElasticNet, it was necessary to obtain a set of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values for the proposed model. These values were obtained by comparing the predictions made by the XGBoost and ElasticNet models to the actual labels of the data.

Once the model had the TP, TN, FP, and FN values, it calculated the following metrics:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}). : \textit{Theproportionofcorrectly classifiedinstancesamonginstances}. \quad (5)$$

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}). : \textit{Theproportionofcorrectlyidentifiedpositiveinstances amongallpositiveinstances}. \quad (6)$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FP}). : \textit{Theproportionofcorrectlyidentifiedpositiveinstances amongallactualpositiveinstances}. \quad (7)$$

$$\text{F1-score} = 2 * (\text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall}). : \textit{Theharmonicmeans ofprecisionandrecall, whichgivesabalancedmeasureofbothmetrics}. \quad (8)$$

Table 5 presents the performance of the ransomware detection model in accurately identifying distinct types of ransomware. The table also shows the model's ability to minimize false positives and false negatives. False positives refer to cases when the model indicates that a system or file has ransomware, when in fact it does not, while false negatives occur when the model fails to detect the presence of ransomware that is there. By providing this information, the dataset enabled us to assess the model's ability to accurately identify diverse types of ransomwares. Furthermore, the dataset includes key performance metrics such as Accuracy, Precision, Recall, and F-Score, which were frequently employed to gauge the effectiveness of the model. These metrics enabled us to compare the effectiveness of various ransomware detection variables in terms of their accuracy in identifying different types of ransomware while minimizing the number of false positives and false negatives.

Table 5. Testing the proposed method on a limited dataset of ransomware anomalies to determine its average performance.

Ransomware	False/Positive	False/Negative	Accuracy	Precision	Recall	F-Score
Kryptik	3.21	1.95	85	0.823	0.853	0.869
Cloud Snooper	1.67	2.84	92	0.882	0.830	0.863
WannaCry	0.95	2.18	81	0.854	0.818	0.861
LockBit	3.53	3.34	88	0.801	0.846	0.832
Black Basta	2.47	1.17	84	0.888	0.839	0.854
Revised Hive	1.92	2.53	89	0.820	0.817	0.829
ALPHV/BlackCat/Noberus	2.99	2.28	95	0.847	0.856	0.844
AvosLocker	1.42	1.11	83	0.897	0.824	0.819
Conti	3.76	3.89	87	0.876	0.811	0.876
REvil	1.08	3.48	80	0.815	0.857	0.877
DarkSide	2.27	1.73	91	0.809	0.814	0.816
Babuk	0.85	3.29	94	0.865	0.847	0.823
Egregor	3.94	3.747	82	0.839	0.819	0.881
Avaddon	2.04	1.09	90	0.896	0.862	0.858

The effectiveness of our feature selection criteria was evaluated by comparing information gain and chi-square methods using the Naïve Bayes classifier. The feature sets were created using 28, 56, 84, 122, and 140 features. Table 6 presents the similarities between the two approaches and their classification performance using four metrics: True Positive, False Positive, Precision, and F-Score. The Bayesian predictor with attributes selected via information gain and chi-square techniques was used to generate the results of the detection process. The results indicated a positive correlation between the number of features used and anomaly detection in both methods, suggesting that accuracy improved when the features were optimized.

Table 6. The level of accuracy achieved by the hybrid "XGBoost and ElasticNet" method in detecting a specific ransomware variant.

Feature Optimization	Applied Feature Count	TP Rate (%)	FP Rate	Precision	F-Score
Information Gain	140	82.96	2.74	0.928	0.844
	112	86.02	2.12	0.867	0.804
	84	81.75	3.14	0.923	0.823
	56	85.14	2.58	0.849	0.862
	28	83	1.98	0.882	0.818
Chi-Square	140	91.94	2.54	0.798	0.844
	112	92.13	3.10	0.826	0.804
	84	91.56	2.28	0.771	0.823
	56	92.01	3.30	0.793	0.862
	28	92.32	1.99	0.810	0.818

We have also used Information Gain as a metric to measure the usefulness of a feature in splitting the data into different classes. It calculated the reduction in entropy achieved by splitting the data on a particular feature. The higher the Information Gain, the more useful the feature is in the classification process. Similarly, Chi-Square is used to determine whether there was a significant association between two categorical variables. In the context of feature selection, Chi-Square was employed to identify features that were significantly associated with the target variable. The higher the Chi-Square score, the more significant the association between the feature and the target variable.

To use these metrics to detect ransomware, we selected the features with the highest Information Gain or Chi-Square score and used them to train the projected model. The selected features were able to distinguish between ransomware and non-ransomware samples with high accuracy. Furthermore, to select the best feature count and metric for detecting ransomware, we compared the TP rate, FP rate, Precision, and F-Score for each combination of feature count and metric. We plotted the results to visualize the performance of each combination.

We have formulated Information Gain as:

$$\text{Information Gain} = \text{Entropy(S)} - \sum[\text{p(v)} \times \text{Entropy(Sv)}] \quad (9)$$

where,

- S is the original dataset.
- v is a specific value of the feature being considered.
- p(v) is the proportion of the number of elements in S that have the value v to the number of elements in S.
- Sv is the subset of S where the feature has the value v.

- Entropy(S) is the entropy of the original dataset S.
- Entropy(Sv) is the entropy of the subset Sv.

The entropy Entropy(S) for the original dataset S is calculated using the following principle:

$$\text{Entropy(S)} = -\sum [\text{p(c)} \times \log 2(\text{p(c)})]. \quad (10)$$

where c is a class label in S, and p(c) is the proportion of the number of elements in S that belong to class c to the number of elements in S. This formula gives the entropy of the original dataset S based on the class labels in the dataset.

Once the entropy of the original dataset S is calculated, we then used the formula for information gain to determine the importance of each feature in S. The information gain measures the reduction in entropy achieved by splitting the data based on a particular feature.

To estimate the Chi-Square, the following formula was employed:

$$\chi^2 = \sum [(O - E)^2 / E] \quad (11)$$

where,

- O is the observed frequency for a given feature and the presence of ransomware.
- E is the expected frequency for the same feature and ransomware presence.
- $\sum$ is the summation over all possible values of the feature and ransomware presence.

The expected frequency was calculated based on the assumption that the feature and the presence of ransomware were independent. If the observed frequency significantly differs from the expected frequency, it suggests that there was a correlation between the feature and the presence of ransomware. In the proposed ransomware detection, Chi-Square was used to identify features that are significantly correlated with ransomware. These features were then used as input for the applied machine learning model (i.e., hybrid XGBoost and ElasticNet) to detect ransomware.

5. Implementation and Testing

Our model was trained using different configurations:

1. The first one involved a Librem 14 laptop equipped with an Intel Core i7 10710U processor with 6 cores and 12 threads, DDR4 RAM of 64 GB, Intel UHD Graphics 620 GPU, M.2 SSD storage of 2 TB (NVMe), PureBoot firmware, and a PureOS operating system.
2. The second configuration used a Librem 5 smartphone, which had an NXP® i.MX 8M Quad core Cortex A53 processor with 64-bit ARM architecture running at a maximum of 1.5 GHz (along with an auxiliary Cortex M4), Vivante GC7000Lite GPU, 3 GB of RAM, 32 GB eMMC internal storage, and a PureOS operating system.

The model was trained with 15,000 instances obtained from an experimental setup. The dataset comprised 27% legitimate instances, 15% crypto miners, 13% memory dumps, 4% RAT-rated files, and 41% ransomware samples, which were customized versions of Kryptik, Cloud Snooper, WannaCry, LockBit, Black Basta, Hive, ALPHV/BlackCat/Noberus, AvosLocker, Conti, REvil, DarkSide, Babuk, and Avaddon. The data were updated as of 23 March 2023. We utilized a simulation model in the VMware NSX sandbox [41] to generate ransomware sample strings. By employing Full-system Mirroring alongside NSX Sandbox, we ensured precise detection capabilities. To gain a deeper understanding of the sandbox's configurations, we additionally executed our script in the Cuckoo Sandbox [42]. This allowed us to observe the behavior of the file within a practical and isolated environment. The decision was made to trust the actual behavior of the files, monitored by the sandbox, allowing us to identify specific features extracted through the sandbox's monitoring.

To identify the necessary prerequisites for a specific action, we employed two separate testing environments. The feature set comprised 140 characteristics, with 30 of them consisting of calls to API packages that encompassed all PureOS application programming interfaces. An outline of the ransomware versions employed in the evaluation stage is provided in Table 7.

Table 7. A high-level view of the analyzed ransomware variants.

Ransomware	Encoding	Lock	Remote Access Trojan	Sample Size (%)
Kryptik	✔	✔	✔	4
Cloud Snooper	✔	✔	✔	9
WannaCry	✔	-	-	7
LockBit	✔	✔	-	5
Black Basta	✔	-	-	11
Revised Hive	✔	✔	-	8
ALPHV/BlackCat/Noberus	✔	-	-	10
AvosLocker	✔	✔	✔	6
Conti	✔	-	✔	12
REvil	✔	-	✔	3
DarkSide	✔	-	✔	8
Babuk	✔	✔	-	2
Egregor	✔	✔	✔	9
Avaddon	✔	✔	-	6

Cuckoo sandbox was capable of analyzing a wide range of file extensions including .js, .hta, .psi, .pdf, .ppt, .ps1, .python, .vbs, .zip, etc. Furthermore, applets, classes (e.g., bin, cpl, dll, etc.), functions (e.g., DllMain, arguments, loader, etc.), dumps (e.g., memory.dump, dump.pcap, tlsmaster.txt, and files.json for metadata extraction), .bson, shots, and more were also examined.

The APIs that were used to facilitate or trigger ransomware operations included a variety of types from various categories. These include, but are not limited to, ShellExecute, CreateProcess, WriteProcessMemory, VirtualAllocEx, RegOpenKey, RegCreateKey, RegSetValue, HttpSendRequest, and LoadLibrary. These APIs belong to a range of different categories such as system calls, networking, input/output, file system, cryptography, and user interface. It is important to note that these APIs were utilized maliciously by threat actors (i.e., pre-fabricated anomalies) to carry out ransomware attacks.

The ransomware we used for encrypting the data (i.e., files of varying sizes ranging from 100KB to 1GB) on the hacked system employed RSA-2048, AES-256, and ChaCha-256 encryption algorithms. We carried out a thorough investigation of the time taken by these algorithms and found that the ChaCha-256 had the fastest encryption speed among the three, making it a more efficient option for use in ransomware attacks. The time-based encryption comparison is shown in Figure 2 and can be summarized as:

(a) Ransomware attackers are using advanced encryption algorithms such as RSA-2048, AES-256, and ChaCha-256 to encrypt victim data, making it inaccessible without the decryption key.
(b) The speed at which encryption algorithms operate can impact the success of a ransomware attack. In this case, ChaCha-256 was found to be the fastest among the three encryption algorithms, making it a potentially more effective choice for attackers.
(c) As a result of the faster encryption speed, ChaCha-256 may become more prevalent in future ransomware attacks.

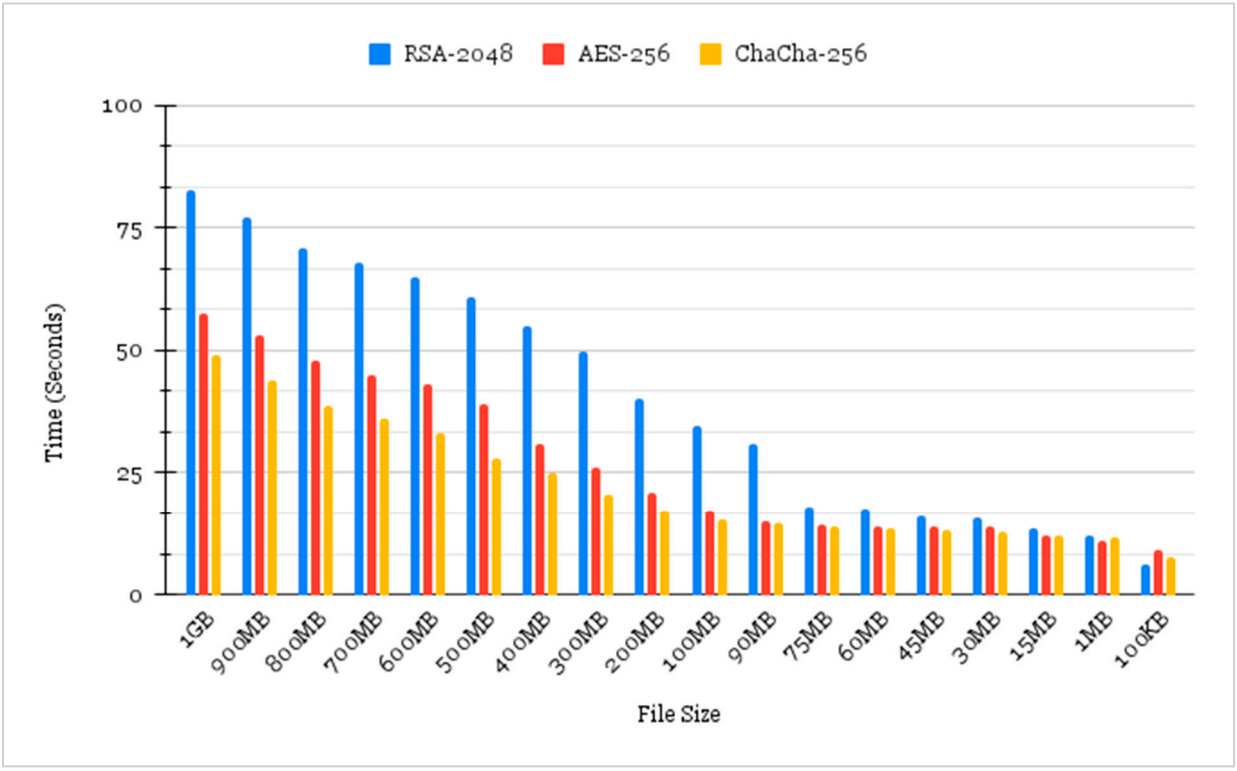

Figure 2. The use of RSA-2048, AES-256, and ChaCha-256 for time-based encryption comparisons.

Figure 3 shows the effect of selecting a subset of features from a dataset of ransomware characteristics based on their variance. The variance threshold is a value that is set to determine the minimum variance a feature must have to be included in the subset. During the optimization process, we noticed that by varying the variance threshold, it is possible to select different numbers of features for the subset. By setting a variance threshold for each feature, only those that significantly differ across the dataset will be included in the subset. If the variance threshold is set high, only features with high variance are included in the subset, leading to a smaller number of ransomware features. In contrast, if the variance threshold is set low, more features with lower variance are included, resulting in a larger number of ransomware features.

It is important to note that the number of ransomware features selected for the subset can have a significant impact on the performance of applied machine learning models (i.e., XGBoost and ElasticNet). Therefore, selecting the optimal number of ransomware features with varying variance thresholds is a crucial step in developing effective ransomware detection and prevention systems.

The study conducted tests on the entire dataset, using a cross-validation technique that involved 25 folds, and splitting the data into training and testing subsets randomly, with 60% of the data used for training and 40% for testing. Table 8 presents the performance of six different machine learning algorithms in detecting and preventing ransomware attacks. The metrics (i.e., accuracy, precision, recall, and F-score) were evaluated using both a 25-fold cross-validation technique and a 60% split training/test set approach. The proposed algorithm, the hybrid XGBoost and ElasticNet, has the highest routine across evaluation techniques, indicating that it outperforms the other algorithms. The results highlight the potential of machine learning algorithms in detecting and preventing ransomware attacks and provide insights into which algorithms perform better in this context.

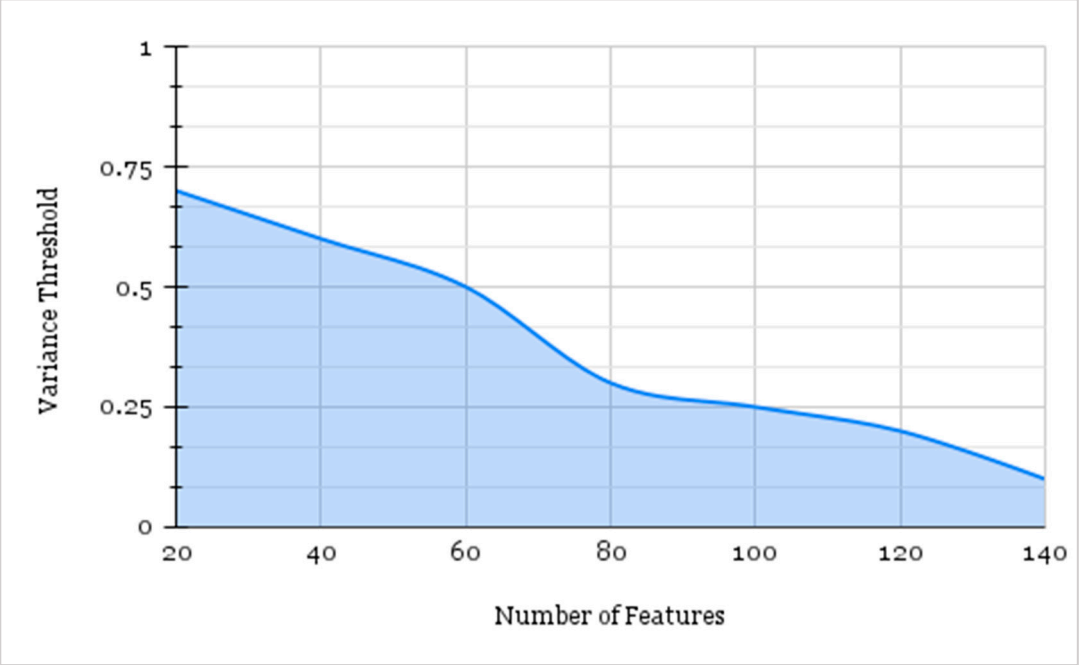

Figure 3. Dimensionality in the number of characteristics with various thresholds for variance.

Table 8. The outcomes of the Accuracy, Precession, Recall, and F-Score.

Sr#	ML Algorithm	Accuracy		Precession		Recall		F-Score	
		25 Folds	60% Split	25 Folds	60% Split	25 Folds	60% Split	25 Folds	60% Split
1	Reinforcement learning (Markovic Decision Process + Q-Learning) [43]	0.867	0.865	0.867	0.865	0.845	0.842	0.874	0.872
2	K-Nearest Neighbors Algorithm [44]	0.872	0.870	0.872	0.871	0.855	0.853	0.880	0.882
3	Support Vector Machine [45]	0.845	0.846	0.846	0.842	0.803	0.806	0.845	0.842
4	Stochastic Gradient Descent [46]	0.811	0.816	0.813	0.817	0.733	0.725	0.804	0.818
5	Naive Bayes [44]	0.512	0.532	0.672	0.666	0.551	0.533	0.865	0.847
6	Hybrid XGBoost and ElasticNet	0.901	0.907	0.921	0.917	0.920	0.933	0.921	0.927

Limitations

The proposed research has primarily focused on exploring the robustness of classifiers that solely examine the structure of binary programs (i.e., of benign and ransomware samples). However, the methods (i.e., hybrid XGBoost and ElasticNet) we have applied would not affect classifiers that consider the execution of such programs, and extract features such as the sequence of system calls. The reason for this limitation is that the data we introduce and modify are not executed during the program runtime. To deal with active features, an attacker would have to resort to using binary rewriting techniques, which are practical modifications specifically designed for this purpose. These alterations involve manipulating the program's anomalous code by adding new branches or replacing semantically equivalent instructions, which can be used to encode ransomware in a way that has broad applicability.

6. Conclusions and Future Work

It can be asserted with a high degree of certainty that machine learning algorithms have proven to be efficacious tools in identifying and detecting malicious software. However, designing such systems is often difficult because they involve complex features that can make it hard to understand how the models learn and accurately identify the real characteristics of malware. Consequently, systems that have these weaknesses can unintentionally incorporate false patterns, which may make them more vulnerable to attacks from malicious actors.

The focal point of this article is the detection of PureOS-specific ransomware, a pernicious and insidious threat that has proliferated with unprecedented velocity in recent years. We conducted a comprehensive examination of the efficacy of various single feature type sets in identifying this type of ransomware, while also considering the customary tactics employed by malevolent actors to camouflage their nefarious activities.

To further refine and optimize these techniques, we exploited hybrid machine learning methodologies, such as XGBoost and ElasticNet, to scrutinize and assess the strength and validity of the developed systems. The ultimate goal was to propose effective methodologies for the judicious implementation of these techniques. Accordingly, experimental work was conducted to evaluate the potential impact of these methodologies on the detection of ransomware and to enhance the design process of hybrid machine-learning-based systems.

It is clear from the results that our approach performs exceptionally well in detecting ransomware patterns with high accuracy and a low false-negative rate. This work shows that ML techniques can be used to significantly improve the effectiveness and efficiency of cybersecurity defenses against ransomware attacks.

We assert that combining multiple machine learning models can improve the overall detection accuracy and reduce false positives. This work might provide a boost to ensemble learning techniques, especially in the area of cyber security.

Author Contributions: Conceptualization, U.T. and T.A.A.; methodology, F.D., U.T. and S.A.C.; software, U.T. and Y.M.; validation, U.T. and T.A.A.; formal analysis, U.T.; investigation, S.A.C. and U.T.; resources, U.T. and T.A.A.; data curation, Y.M. and U.T.; writing—original draft preparation, T.A.A., U.T. and F.D.; writing—review and editing, Y.M., U.T. and S.A.C..; visualization, Y.M. and U.T.; supervision, T.A.A.; project administration, S.A.C. and Y.M.; funding acquisition, S.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through project number 2022/01/22636.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, Saudi Arabia.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This study is supported via funding from Prince Sattam Bin Abdulaziz University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lawal, K.; Rafsanjani, H.N. Trends, benefits, risks, and challenges of IoT implementation in residential and commercial buildings. *Energy Built Environ.* **2022**, *3*, 251–266. [CrossRef]
2. Ransomware at Colorado IT Provider Affects 100+ Dental Offices—Krebs on Security. 7 December 2019. Available online: https://krebsonsecurity.com/2019/12/ransomware-at-colorado-it-provider-affects-100-dental-offices/ (accessed on 27 March 2023).
3. NATO Countries Hit with Unprecedented Cyber Attacks. GovTech. 4 September 2022. Available online: https://www.govtech.com/blogs/lohrmann-on-cybersecurity/nato-countries-hit-with-unprecedented-cyber-attacks (accessed on 28 March 2023).
4. Cui, J. Malware Detection Algorithm for Wireless Sensor Networks in a Smart City Based on Random Forest. *J. Test. Eval.* **2022**, *51*, 20220100. [CrossRef]

5. Singh, T.; Di Troia, F.; Corrado, V.A.; Austin, T.H.; Stamp, M. Support Vector Machines and Malware Detection. *J. Comput. Virol. Hacking Tech.* **2015**, *12*, 203–212. [CrossRef]
6. Yilmaz, A.B.; Taspinar, Y.S.; Koklu, M. Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms. *Int. J. Intell. Syst. Appl. Eng.* **2022**, *10*, 269–274. Available online: https://ijisae.org/index.php/IJISAE/article/view/2010 (accessed on 29 March 2023).
7. Abu Al-Haija, Q.; Odeh, A.; Qattous, H. PDF Malware Detection Based on Optimizable Decision Trees. *Electronics* **2022**, *11*, 3142. [CrossRef]
8. Gao, Y.; Hasegawa, H.; Yamaguchi, Y.; Shimada, H. Malware Detection Using LightGBM with a Custom Logistic Loss Function. *IEEE Access* **2022**, *10*, 47792–47804. [CrossRef]
9. Xie, N. Andro_MD: Android Malware Detection based on Convolutional Neural Networks. *Int. J. Perform. Eng.* **2018**, *14*, 547–558. [CrossRef]
10. Liu, T.; Li, Z.; Long, H.; Bilal, A. NT-GNN: Network Traffic Graph for 5G Mobile IoT Android Malware Detection. *Electronics* **2023**, *12*, 789. [CrossRef]
11. Manoharan, S.; Sugumaran, P.; Kumar, K. Multichannel Based IoT Malware Detection System Using System Calls and Opcode Sequences. *Int. Arab. J. Inf. Technol.* **2022**, *19*, 261–271. [CrossRef]
12. Sun, H.; Wang, X.; Buyya, R.; Su, J. CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained internet of things (IoT) devices. *Softw. Pract. Exp.* **2016**, *47*, 421–441. [CrossRef]
13. Ahmed, U.; Lin, J.C.W.; Srivastava, G. Mitigating adversarial evasion attacks of ransomware using ensemble learning. *Comput. Electr. Eng.* **2022**, *100*, 107903. [CrossRef]
14. Ibrahim, A.; Tariq, U.; Ahamed Ahanger, T.; Tariq, B.; Gebali, F. Retaliation against Ransomware in Cloud-Enabled PureOS System. *Mathematics* **2023**, *11*, 249. [CrossRef]
15. Barrett, M.P. Framework for Improving Critical Infrastructure Cybersecurity Version 1.1. NIST. 16 April 2018. Available online: https://www.nist.gov/publications/framework-improving-critical-infrastructure-cybersecurity-version-11 (accessed on 27 March 2023).
16. Hull, G.; Jhon, H.; Arief, B. Ransomware deployment methods and analysis: Views from a predictive model and human responses. *Crime Sci.* **2019**, *8*, 2. [CrossRef]
17. Kharraz, A.; Robertson, W.; Kirda, E. Protecting against Ransomware: A New Line of Research or Restating Classic Ideas? *IEEE Secur. Priv.* **2018**, *16*, 103–107. [CrossRef]
18. Upadhyaya, R.; Jain, A. Cyber ethics and cyber crime: A deep dwelved study into legality, ransomware, underground web and bitcoin wallet. In Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 29–30 April 2016; pp. 143–148. [CrossRef]
19. Gagneja, K.K. Knowing the ransomware and building defense against it—Specific to healthcare institutes. In Proceedings of the 2017 Third International Conference on Mobile and Secure Services (MobiSecServ), Miami Beach, FL, USA, 11–12 February 2017; pp. 1–5. [CrossRef]
20. Celdrán, A.H.; Sánchez, P.M.S.; Castillo, M.A.; Bovet, G.; Pérez, G.M.; Stiller, B. Intelligent and behavioral-based detection of malware in IoT spectrum sensors. *Int. J. Inf. Secur.* **2022**, *22*, 541–561. [CrossRef]
21. Moon, D.; Lee, J.; Yoon, M. Compact feature hashing for machine learning based malware detection. *ICT Express* **2022**, *8*, 124–129. [CrossRef]
22. Dargahi, T.; Dehghantanha, A.; Bahrami, P.N.; Conti, M.; Bianchi, G.; Benedetto, L. A Cyber-Kill-Chain based taxonomy of crypto-ransomware features. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 277–305. [CrossRef]
23. ESET: Threat Report Q2 2020. *Comput. Fraud. Secur.* **2020**, *2020*, 4. [CrossRef]
24. Yang, W.; Gao, M.; Chen, L.; Liu, Z.; Ying, L. RecMaL: Rectify the malware family label via hybrid analysis. *Comput. Secur.* **2023**, *128*, 103177. [CrossRef]
25. VirusChaser: A Comprehensive Antivirus Solution Equipped with Powerful System Protection Features. VirusChaser. 18 February 2023. Available online: https://www.ncloud.com/marketplace/viruschaser (accessed on 16 April 2023).
26. FKIE, F. Cloud Snooper (Malware Family). Cloud Snooper (Malware Family). 21 December 2020. Available online: https://malpedia.caad.fkie.fraunhofer.de/details/elf.cloud_snooper (accessed on 1 March 2023).
27. Tonido—Run Your Personal Cloud. A Free Private Cloud Server. (n.d.). Tonido—Run Your Personal Cloud. A Free Private Cloud Server. Available online: https://www.tonido.com/ (accessed on 2 March 2023).
28. Ghafur, S.; Kristensen, S.; Honeyford, K.; Martin, G.; Darzi, A.; Aylin, P. A retrospective impact analysis of the WannaCry cyberattack on the NHS. *npj Digit. Med.* **2019**, *2*, 98. [CrossRef]
29. Eliando, E.; Purnomo, Y. LockBit 2.0 Ransomware: Analysis of infection, persistence, prevention mechanism. *CogITo Smart J.* **2022**, *8*, 232–243. [CrossRef]
30. Kajave, A.; Nismy, S.A.H. How Cyber Criminal Use Social Engineering to Target Organizations. *arXiv* **2022**, arXiv:2212.12309. [CrossRef]
31. Tanner, D.A.; Hinchliffe, A.; Santos, D. Threat Assessment: Blackcat Ransomware. 2022. Available online: https://shorturl.at/cdV37 (accessed on 2 March 2023).
32. Kara, I.; Aydos, M. The rise of ransomware: Forensic analysis for windows based ransomware attacks. *Expert Syst. Appl.* **2022**, *190*, 116198. [CrossRef]

33. Umar, R.; Riadi, I.; Kusuma, R.S. Analysis of Conti Ransomware Attack on Computer Network with Live Forensic Method. *IJID Int. J. Inform. Dev.* **2021**, *10*, 53–61. [CrossRef]
34. Datta, P.M.; Acton, T. From disruption to ransomware: Lessons from hackers. *J. Inf. Technol. Teach. Cases* **2022**. [CrossRef]
35. Purism Products. Available online: https://puri.sm/products/ (accessed on 3 March 2023).
36. Zou, S.; Zhang, J.; Jiang, S.; Cheng, Y.; Ji, X.; Xu, W. OutletGuarder: Detecting DarkSide Ransomware by Power Factor Correction Signals in an Electrical Outlet. In Proceedings of the 2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS), Nanjing, China, 10–12 January 2023; pp. 419–426. [CrossRef]
37. Lin, C.; Kimberly, G.; Daniel, R.; Henry, U. Blockchain Forensics and Crypto-Related Cybercrimes. *SSRN* **2023**. [CrossRef]
38. PureBoot & Ndash; Purism. (n.d.). Purism. Available online: https://puri.sm/projects/pureboot/ (accessed on 1 March 2023).
39. Palša, J.; Ádám, N.; Hurtuk, J.; Chovancová, E.; Madoš, B.; Chovanec, M.; Kocan, S. MLMD—A Malware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm. *Appl. Sci.* **2022**, *12*, 6672. [CrossRef]
40. Srinivasan, S.; Deepalakshmi, P. ENetRM: ElasticNet Regression Model based malicious cyber-attacks prediction in real-time server. *Meas. Sens.* **2023**, *25*, 100654. [CrossRef]
41. VMware. NSX Sandbox | VMware. Available online: https://www.vmware.com/products/nsx-sandbox.html (accessed on 4 March 2023).
42. Wahidin, G.W.; Syaifuddin, S.; Sari, Z. Analisis Ransomware Wannacry Menggunakan Aplikasi Cuckoo Sandbox. *J. Repos.* **2022**, *4*, 83–94. [CrossRef]
43. Lee, C.; Han, S.M.; Chae, Y.H.; Seong, P.H. Development of a cyberattack response planning method for nuclear power plants by using the Markov decision process model. *Ann. Nucl. Energy* **2022**, *166*, 108725. [CrossRef]
44. Sahin, D.O.; Akleylek, S.; Kilic, E. LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers. *IEEE Access* **2022**, *10*, 14246–14259. [CrossRef]
45. Singh, P.; Borgohain, S.K.; Kumar, J. Performance Enhancement of SVM-based ML Malware Detection Model Using Data Preprocessing. In Proceedings of the 2022 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), Patna, India, 24–25 June 2022; pp. 1–4. [CrossRef]
46. Mowri, R.A.; Siddula, M.; Roy, K. Interpretable Machine Learning for Detection and Classification of Ransomware Families Based on API Calls. *arXiv* **2022**, arXiv:2210.11235. [CrossRef]