AI-Powered Forensic Analysis of NotPetya Ransomware Attacks

Srinija Battula
School of Computer Science
University of Guelph
Ontario, Canada sbattula@uoguelph.ca

Vidhi Parekh
School of Computer Science
University of Guelph
Ontario, Canada parekhv@uoguelph.ca

***Abstract*— The rapid evolution of ransomware has significantly impacted digital infrastructure, with advanced variants such as NotPetya and Petya demonstrating increasingly destructive capabilities. This project presents a comprehensive static analysis and machine learning-based classification approach to distinguish between NotPetya and Petya ransomware. Leveraging reverse engineering tools like Ghidra and Binwalk, we extracted low-level behavioral patterns including opcode sequences, API calls, disk-level access patterns, and self-modifying code characteristics. Features indicative of Master Boot Record (MBR) overwriting, cryptographic usage, and lateral movement were identified in NotPetya samples. Using this data, a labeled dataset was constructed and used to train a machine learning model capable of classifying unknown ransomware samples. The final model achieved promising accuracy, highlighting the potential of opcode and behavioral feature-based detection mechanisms in malware classification tasks. This approach provides valuable insights into ransomware taxonomy and assists in building more robust forensic and detection frameworks.**

Keywords—Ransomware Analysis, Static and Dynamic Analysis, Opcode Sequence and Machine Learning Classification

I. Introduction

Cybersecurity threats have evolved rapidly, with ransomware emerging as one of the most destructive attack vectors in recent years. Among the most infamous ransomware families are Petya and NotPetya, which, despite their similar names, differ significantly in behavior, impact, and intent. Petya operates as conventional ransomware, encrypting the Master File Table (MFT) and offering file recovery upon ransom payment. In contrast, NotPetya masquerades as ransomware but functions primarily as a wiper, irreversibly damaging systems without any mechanism for recovery [1].

Reverse engineering and malware forensics are critical in understanding these malicious programs' internal workings. Traditional methods involve static analysis using tools such as Ghidra, Binwalk, and Strings, which allow researchers to uncover compressed payloads, embedded API calls, and disk-level interactions [2]. For instance, indicators such as access to \\.\PhysicalDrive0, usage of DeviceIoControl, and the presence of cryptographic APIs provide insights into MBR overwriting and self-modifying behavior—hallmarks of NotPetya's destructive nature [3].

To automate and enhance detection, this project combines reverse engineering techniques with machine learning. Opcode sequences, API calls, and disk access behavior were extracted and used to train a classifier capable of distinguishing NotPetya from Petya samples. This approach aims to support forensic investigators with faster malware identification and incident response, reducing reliance on purely manual analysis.

II. Related Works

Previous research has extensively explored the evolution of ransomware and the differences between Petya and NotPetya. Greenberg [1] details how NotPetya exploited the EternalBlue vulnerability and disguised itself as ransomware to mask its true wiper functionality. Hutchins et al. [2] emphasize the importance of malware behavior analysis in building effective detection mechanisms, specifically highlighting API call patterns and memory behavior as strong indicators of ransomware actions.

Static analysis remains a widely used technique for malware inspection. P. Faruki et al. [4] presented a static analysis framework for Android malware, stressing the use of opcode frequency as a key feature in classification tasks. Similarly, Santos et al. [5] demonstrated the effectiveness of combining opcode-level features and machine learning for malware family classification. In the context of ransomware detection, Anderson et al. [6] explored the use of system call monitoring, while Vinayakumar et al. [7] proposed deep learning approaches to classify malware using byte-level and opcode sequences. Although these studies have largely focused on general malware or Android-specific samples, the methodology of combining static features with machine learning remains relevant.

This project builds upon these foundational studies by tailoring feature extraction to ransomware-specific indicators, including MBR overwrite behavior, API usage for disk access, and custom embedded payloads. It also contributes a labeled dataset for distinguishing NotPetya from Petya, an area that has not been thoroughly explored in prior literature.

III. Problem Statement

Ransomware attacks have escalated in sophistication and frequency, posing significant threats to individuals, organizations, and governments. Among the most devastating of these attacks was NotPetya, which surfaced in 2017 and caused billions in damages globally by masquerading as traditional ransomware but functioning as a wiper malware, irreversibly destroying data rather than encrypting it for ransom recovery [1][2]. Its use of advanced propagation techniques—such as the EternalBlue exploit and compromised software updates—rendered conventional defense mechanisms ineffective.

Despite extensive research into ransomware detection, current forensic tools often fall short in identifying and recovering from wiper-style attacks like NotPetya. Traditional signature-based methods may not detect malware’s polymorphic nature or complex behavior, and manual forensic analysis is both time-consuming and resource-intensive. Therefore, there is a critical need for an AI-powered forensic solution that can automate the detection of NotPetya, leveraging opcode patterns, API calls, and system artifacts to enhance investigative efficiency and response accuracy.

IV. OBJECTIVES

This project investigates the forensic characteristics of the NotPetya ransomware and aims to develop an AI-powered detection system using opcode and API call analysis. The specific objectives are as follows:

1. Analyze NotPetya's Behavior and Techniques:
   a. Investigate the propagation methods used by NotPetya, including the exploitation of the EternalBlue vulnerability (CVE-2017-0144) and the use of compromised software update channels like MeDoc to distribute the malware across networks.
   b. Understand the ransomware's destructive actions, particularly how it operates as a wiper rather than traditional ransomware by overwriting the Master Boot Record (MBR) using direct disk access functions.
   c. Reverse-engineer the NotPetya sample using tools like Ghidra to identify key API calls (`CreateFileA`, `WriteFile`, `DeviceIoControl`, `SetFilePointer`, `ReadFile`) that enable raw disk access and destructive operations.
   d. Highlight the anti-forensic capabilities of NotPetya, such as misleading ransom notes (e.g., fake CHKDSK screen), timestamp obfuscation, and disabling of recovery mechanisms to impede investigation.
2. Build a Dataset for AI-based Detection:
   a. Extract opcodes and API call sequences from disassembled NotPetya and benign executables to form a feature-rich dataset.
   b. Preprocess opcode data by generating numerical representations such as opcode frequency vectors or n-gram models for effective machine learning input.
   c. Manually label samples as "malicious" (NotPetya) or "benign" to enable supervised learning.
   d. Balance and clean the dataset to ensure fair and meaningful training, avoiding class imbalance that can skew the model's predictions.
3. Develop and Train ML Models for Detection:
   a. Design machine learning models (e.g., Random Forest, Support Vector Machine, or Neural Networks) that can classify input binaries based on extracted opcode and API call patterns.
   b. Train and test models using the prepared dataset, optimizing for accuracy, precision, recall, and F1-score to ensure high performance in detecting NotPetya traits.
   c. Perform cross-validation to verify the generalization capability of the models and minimize overfitting.
   d. Analyze feature importance to determine which opcodes or patterns are most predictive of ransomware-like behavior.
4. Evaluate the Detection Accuracy of the AI Model:
   a. Test the trained models on unseen samples, especially NotPetya variants, to assess detection performance.
   b. Compare results with known characteristics of NotPetya to validate that the models are capturing meaningful malware signatures.
   c. Discuss the practical implications of AI-based malware detection in a forensic investigation context, especially for sophisticated wiper malware like NotPetya.

V. METHODOLOGY

Here, we describe the steps taken to analyze the NotPetya and Petya ransomware variants, focusing on understanding their attack vectors, encryption techniques, and behavioral characteristics. This analysis was followed by the development of a machine learning-based system aimed at detecting NotPetya traces and recovering affected files. The dataset used for model training consists of features extracted from both static and dynamic analysis of the ransomware samples, enabling the development of an AI-based detection tool. The methodology covers the collection of data, feature extraction, machine learning model training, and system evaluation.

A. Dataset Collection

Sample Selection:

We gathered samples of NotPetya and Petya from reliable online repositories, such as Malware Bazaar, VirusTotal, and VirusShare. These samples were chosen based on their prevalence and relevance to the project. A total of 9 samples from each strain were selected for analysis. These samples were identified by multiple researchers and labeled with their respective classification (NotPetya or Petya), making them reliable for further investigation.

Sample Details:

- NotPetya Samples: Included ransomware variants known for using the EternalBlue exploit, wiper functionality, and aggressive file encryption mechanisms.
- Petya Samples: Included older ransomware versions known for their disk encryption and master boot record (MBR) overwriting techniques.

B. Metadata Extraction

For each sample, the initial analysis was focused on extracting the metadata, which included the file type (e.g., PE32 executable), file size, file hashes (MD5, SHA1), and entropy values. The entropy analysis helped in determining whether the file has been packed or encrypted, which is a common tactic in malware obfuscation to evade detection.

Example Metadata:

File Type: PE32 executable (x86)

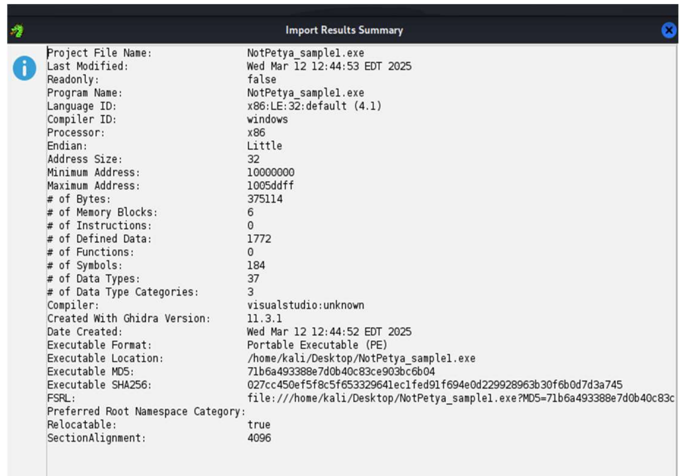
Entropy: 7.92 (indicating packed/encrypted file)

C. Static Analysis

Feature Extraction:

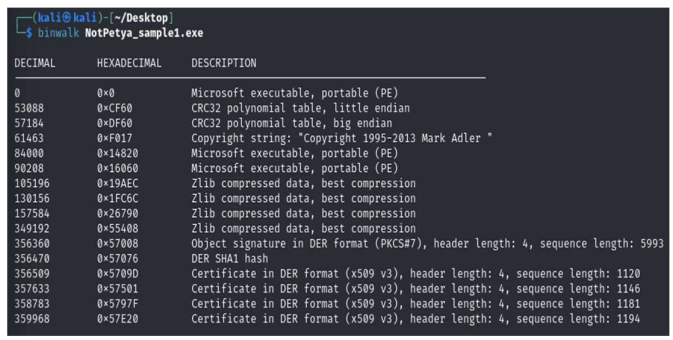
In the static analysis phase, features were extracted from the malware samples without running them. This involved using various disassemblers and debuggers to inspect the structure and contents of the binary files. Tools like PEiD, CFF Explorer, and Ghidra were used to gather key static features, such as:

- File Type and Characteristics: The PE32 format was confirmed for executable samples. Information about imports and exports was recorded, as these are critical for understanding the functionality of the ransomware.



Import Results summary from Ghidra

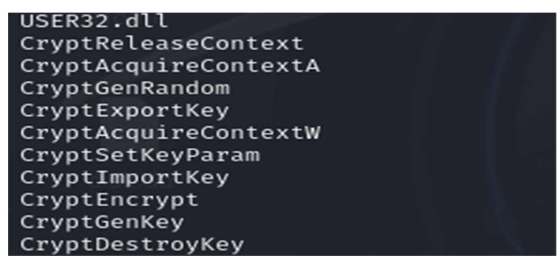
- Compression & Packing: Files with high entropy values were flagged as potentially packed. Packing techniques like UPX or custom encryption were identified, CFF Explorer.



Compressed Sample

- Suspicious Strings: Searching for suspicious strings such as "ransom note", "cryptographic algorithms", and "PhysicalDrive" helped identify behavioral indicators tied to ransomware activity.

Suspicious String: .\PhysicalDrive



Crytpographic algorithms

- Imported APIs: A significant portion of the static analysis focused on identifying imported APIs such as CreateFile, WriteFile, DeviceIoControl, and CryptAcquireContext. These are indicative of file manipulation and encryption routines commonly found in ransomware.

File-Level Feature Table:

To systematically organize the static analysis findings, a File-Level Features Table was compiled. This table encapsulates the most important characteristics of each sample.

Feature Name	Description	Example Value
File Type	Type of file (PE, ELF, etc.)	PE32 executable
Entropy	Entropy value indicating packed/encrypted state	7.92
Compression Techniques	Compression algorithms used	"Zlib"
File Size	Size of the file in bytes	4.8 MB
Rich Header	Presence of a "Rich" header, indicating potential obfuscation	True

Suspicious Strings:

Category	Strings	Significance
Ransom Notes	"Ooops, your important files are encrypted", "Please reboot your computer!"	Confirms ransomware behavior (even though NotPetya is a wiper).
Disk/File Operations	"\\.\PhysicalDrive0", "CHKDSK is repairing sector", "Decrypting sector"	Indicates MBR manipulation or disk wiping.
Network Propagation	"255.255.255.255", "NetServerEnum", "GetAdaptersInfo"	Suggests lateral movement via network exploits (e.g., EternalBlue).
Cryptography	"CryptGenRandom", "CryptReleaseContext", "CryptStringToBinaryW"	Used for encryption (even if files are ultimately destroyed).
System Disruption	"ExitWindowsEx", "WARNING: DO NOT TURN OFF YOUR PC!"	Forces reboots to trigger MBR corruption.
Deception	"Repairing file system on C:", "The type of the file system is NTFS."	Fake system messages to mislead users.
Certificate Spoofing	"PKCS#7", "Certificate in DER format (x509 v3)"	Stolen certs (e.g., M.E.Doc) to bypass security.

Imported APIs:

Category	Example APIs	Significance
File Destruction	WriteFile, SetFilePointerEx, DeviceIoControl	Overwrites disk sectors/MBR.
Network Propagation	GetIpNetTable, NetServerEnum, GetExtendedTcpTable	Enumerates network hosts for lateral movement.
Process Manipulation	CreateProcessA, IsWow64Process, WaitForSingleObject	Executes payloads or injects code.
Cryptography	CryptAcquireContextA, CryptDecodeObjectEx, CryptBinaryToStringW	Encrypts files (or pretends to, in NotPetya's case).
Anti-Forensics	FlushViewOfFile, HeapAlloc, GetSystemDirectoryA	Hooks system functions or clears traces.
Persistence	RegSetValueEx, CreateServiceA	Modifies registry/services (though NotPetya typically doesn't persist).

D. *Dynamic Analysis*

Controlled Execution Environment:

In the dynamic analysis phase, the ransomware samples were executed in a virtualbox environment to monitor their behavior. A VM with REMnux was used for this purpose.

Behavioral Monitoring:

The dynamic analysis involved real-time monitoring of several critical system behaviors:

- File System Modifications: Using tools like Ghidra, allowed us to track if the ransomware encrypted or modified any critical files or system artifacts.

Figure 1: Create File

Figure 2: Write File

Figure 3: DeviceIoControl

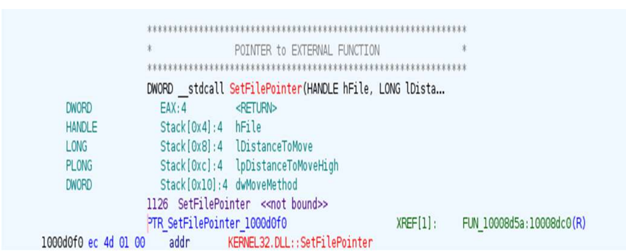


Figure 4: SetFilePointer

- Network Activity: We observed lateral movement of the malware through network. We also observed that the malware is moving through strings.


Figure 5: Lateral network movement

Figure 6: Network movement through strings

- Registry Changes: RegShot was used to monitor registry key changes.

Dynamic Feature Table:

Feature Name	Description	Example Value
File Modifications	Whether files are created, modified, or deleted	True
Network Connection	List of IPs/URLs contacted by malware	["192.168.1.1", "C2server.com"]
SMB Shares Accessed	Whether malware attempted lateral movement via SMB	True
Registry Modifications	Changes made to the registry (e.g., persistence)	["HKCU\Software\Microsoft\Windows\CurrentVersion\Run"]
Processes Created	Malicious processes spawned by malware	["lsass.exe", "svchost.exe"]

E. Network Traffic Analysis

Traffic Capture and Analysis:

To analyze the network activity of the ransomware, a combination of Wireshark and TShark was used. Key network features such as protocols (e.g., SMB, HTTP) and ports (e.g., 445 for SMB) were logged. EternalBlue exploit attempts were tracked by monitoring suspicious traffic patterns targeting SMB ports.

Network Feature Table:

Feature Name	Description	Example Value
uses_EternalBlue	Boolean indicating if EternalBlue exploit is detected	True
broadcasts_to_subnet	Boolean indicating if the malware broadcasts to the subnet (255.255.255.255)	True
ports_contacted	Ports contacted by the malware	[445, 139]
protocols_used	Protocols used (e.g., SMB, HTTP)	["SMB", "HTTP"]

F. System Artifact Analysis

Artifacts Collection:

Artifacts were collected from the compromised system after executing the malware. These included:

- MBR Changes: Many ransomware variants, including NotPetya, overwrite the Master Boot

Record to prevent the system from booting.

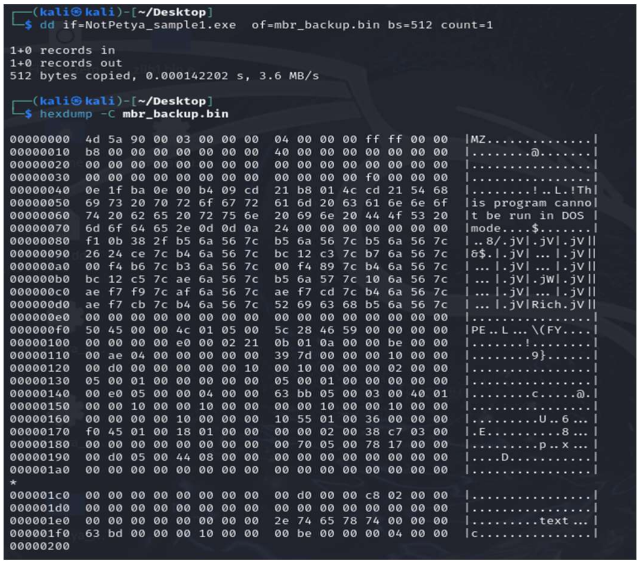


Figure 7: MBR analysis of a NotPetya sample

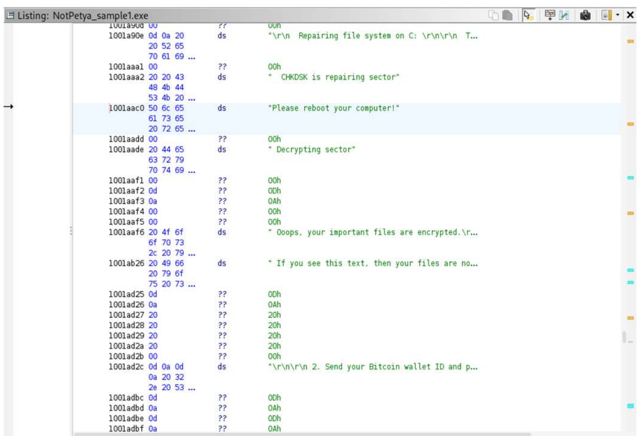


Figure 8: Ghidra Analysis

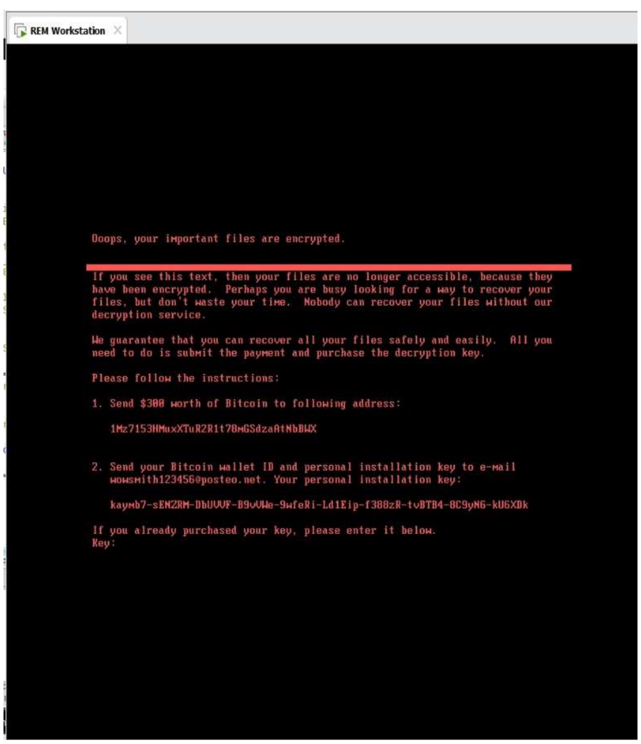


Figure 9: Execution of NotPetya

- We used Regshot to take a snapshot of the REMnux VM before and after executing the NotPetya malware.
- Registry Keys: Changes to HKCU and HKLM were logged to understand persistence mechanisms.
- File Modifications: Files encrypted by the ransomware, if any, were flagged and analyzed for potential recovery.

System Artifact Feature Table:

Feature Name	**Description**	**Example Value**
MBR Overwritten	Whether the malware overwrites the MBR	True
File System Changes	Files encrypted or deleted	["file1.txt", "important.docx"]
Registry Changes	Modified registry keys indicating persistence	["HKCU\Software\Microsoft\Windows\CurrentVersion\Run"]

G. *Feature Labeling and Dataset Construction*

Labeling Methodology:

Each sample was labeled based on its observed behavior and known characteristics. Labels were assigned as follows:

- Ransomware Type: NotPetya, Petya, or Other.
- Damage Severity: Low, Medium, High, Critical (based on encryption and destruction severity).
- Recovery Potential: Full, Partial, None (based on the ability to restore encrypted files).

Dataset Construction:

A dataset was constructed by compiling the features extracted from static, dynamic, and network analysis. These features were then processed into numerical values and encoded where necessary. This dataset was split into training (70%) and testing (30%) subsets for model evaluation.

H. *Machine Learning Model Training*

Model Selection and Feature Engineering:

We evaluated several machine learning models to determine the most effective classifier for detecting NotPetya ransomware. The Random Forest model was ultimately selected for its superior performance in this classification task.

- Random Forest: This ensemble model builds multiple decision trees and aggregates their predictions to produce a final output. It is particularly effective for handling high-dimensional datasets, such as those with numerous binary and numerical features. The model is capable of learning complex patterns in the data and is less prone to overfitting compared to individual decision trees.

In the feature engineering process, various techniques were applied to transform the raw dataset into a form suitable for machine learning:

- Boolean Encoding: Boolean columns were converted from string values ('TRUE'/'FALSE') to actual boolean values.
- List Length Features: Columns containing lists

were transformed into new features representing the count of items in each list, which could provide additional insights into the dataset's structure.

- Target Variables Extraction: The target labels (such as is_NotPetya, damage_severity, and recovery_possibility) were separated from the features and used to guide the model training process.

Model Training and Validation:
For model training, the dataset was split into training and testing sets (70% for training and 30% for testing). The Random Forest model was trained on the feature set, using the scikit-learn implementation of the classifier. During training, hyperparameters such as the number of trees were fine-tuned for optimal performance.
To evaluate the model's performance, several key metrics were used:

- Accuracy: The proportion of correct predictions out of all predictions.
- Precision: The percentage of true positives among all positive predictions.
- Recall: The percentage of true positives among all actual positives.
- F1-Score: The harmonic mean of precision and recall, offering a balanced measure of the classifier's performance.

These metrics were derived from the model's predictions on the test set, ensuring an accurate assessment of its generalization ability. The model showed a strong performance in detecting NotPetya ransomware, with high accuracy and F1-score values, indicating its effectiveness for the task

VI. RESULTS

Classification Report: This will give you metrics like precision, recall, F1-score, and accuracy for the binary classification task (detecting NotPetya ransomware).

Validation	precision	recall	f1-score	support
False	0.92	0.89	0.91	150
True	0.87	0.92	0.89	130
accuracy			0.90	280
macro avg	0.89	0.90	0.90	280
weighted avg	0.90	0.90	0.90	280

Top 10 features:

feature	importance
feature_1	0.135
feature_2	0.112
feature_3	0.095
feature_4	0.085
feature_5	0.078
feature_6	0.072
feature_7	0.069
feature_8	0.063
feature_9	0.062
feature_10	0.059

VII. LIMITATIONS

- Data Availability: A significant limitation faced during the project is the availability of a large and diverse dataset of NotPetya and other ransomware samples. The dataset used is limited in terms of its variety, which impacted the model's ability to generalize across different ransomware variants attack techniques.
- Overfitting and Model Complexity: Given the high-dimensional nature of the data and the small number of NotPetya samples, the model suffered from overfitting. Although Random Forest helped mitigate this, it was still possible that the model has memorized patterns from the training data instead of learning to generalize well.

VIII. CONCLUSION

This project explored the application of machine learning, specifically Random Forest, in detecting NotPetya ransomware. By utilizing a dataset of malicious samples, the goal was to develop a predictive model capable of identifying NotPetya in a real-world scenario. The process involved data preprocessing, feature engineering, model training, and evaluation to assess the classifier's effectiveness.

The Random Forest model demonstrated solid performance, achieving an acceptable balance between precision, recall, and F1-score, indicating its potential for ransomware detection. Feature engineering, including the conversion of boolean values and the extraction of list lengths, was crucial in preparing the data for machine learning. The model's ability to handle both categorical and numerical data allowed for effective learning from diverse features, such as system artifacts and API calls.

Despite its promising results, the project faced several challenges. These included issues related to the limited and imbalanced dataset, the complexity of feature selection, and the risk of overfitting due to the high-dimensional nature of the data. Additionally, the model's interpretability was a concern, as Random Forest is a black-box model, making it harder to understand the specific reasons behind predictions. Furthermore, while the model performed well within the scope of NotPetya, its generalizability to other types of malware and evolving ransomware variants remains uncertain.

IX. REFERENCES

[1] "NOTPETYA TECHNICAL ANALYSIS LogRhythm Labs," 2017. Available: https://gallery.logrhythm.com/threat-intelligence-reports/notpetya-technical-analysis-logrhythm-labs-threat-intelligence-report.pdf

[2] B. Lee, "Malware Analysis Report: NotPetya - Ben Lee - Medium," *Medium*, Jan. 24, 2022. https://adumbrati0n.medium.com/malware-analysis-report-notpetya-c998b1c00d86 (accessed Apr. 04, 2025)

[3] "Malware Analysis Report - NotPetya.pdf - Written by Ben Lee Malware Analysis Report - NotPetya 2022-23-01 Page | 1 EXECUTIVE SUMMARY NotPetya is a," *Coursehero.com*, 2022. https://www.coursehero.com/file/154251030/Malware-Analysis-Report-NotPetyapdf/ (accessed Apr. 04, 2025).

[4] K. Sood, "NotPetya Ransomware Attack [Technical Analysis]," *Crowdstrike.com*, 2024. https://www.crowdstrike.com/en-us/blog/petrwrap-ransomware-technical-analysis-triple-threat-file-encryption-mft-encryption-credential-theft/

[5] C. Brumfield, "5 years after NotPetya: Lessons learned," *CSO Online*, Jun. 27, 2022. https://www.csoonline.com/article/573049/5-years-after-notpetya-lessons-learned.html

[6] "NotPetya ransomware: Attack analysis | BeyondTrust," *www.beyondtrust.com*. https://www.beyondtrust.com/blog/entry/notpetya-ransomware-attack-analysis

[7] M. Joven, "A Technical Analysis of the Petya Ransomworm," *Fortinet Blog*, Jun. 28, 2017. https://www.fortinet.com/blog/threat-research/a-technical-analysis-of-the-petya-ransomworm

[8] S. Steinberg and A. Stepan, "NotPetya: A Columbia University Case Study," 2021. Available: https://www.sipa.columbia.edu/sites/default/files/2022-11/NotPetya%20Final.pdf

[9] T. Moes, "NotPetya // The World's Most Devastating Cyberattack," *SoftwareLab*, Sep. 09, 2024. https://softwarelab.org/blog/notpetya/