Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Network-Centric Ransomware Intrusion Detection Framework

Gervais Hatungimana

G.Hatungimana@latrobe.edu.au

La Trobe University

Abdun Naser Mahmood

La Trobe University

Mohammad Jabed Morshed Chowdhury

La Trobe University

Research Article

Keywords:

Posted Date: November 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11427282/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Network-Centric Crypto-Ransomware Intrusion Detection Framework

Gervais Hatungimana (corresponding author)[a,1], **Abdun Naser Mahmood**[b,1], **Mohammad Jabed Morshed Chowdhury**[c,1]

[1]La Trobe University, Melbourne, Victoria, Australia

Received: date / Accepted: date

Abstract For decades, ransomware has remained one of major cybersecurity concerns. While defensive solutions can prevent the delivery and execution of malicious payloads, they can often be bypassed through evasive techniques. From a post-compromise perspective, network-layer telemetry represents the final source of visibility for detecting and containing ongoing intrusions. Despite its strategic importance, machine learning-based ransomware detection using network traffic behavior has received comparatively limited attention in recent research [1]. Recently proposed methods exhibit limitations related to behavioral overlap between benign and malicious activities [2] and limited observability of ransomware operations [3]. To address these limitations, this paper makes three key contributions including the introduction of **Traffic of Interest (ToI)**, a novel network traffic analysis technique for ransomware detection; development of **a new behavioural dataset** containing a broader set of ransomware families and ransomware-like benign behaviours; and the identification of **novel behavioural Indicators of Compromise (IoCs)**. The experimental results demonstrate good performance of the proposed machine learning framework, achieving 99.64% precision, a 0% false negative rate (FNR), and a minimal false positive rate (FPR) for normal detection, as well as 99.44% accuracy for early detection. Beyond machine learning-based detection, the study also produces a repository of behavioural network-based Indicators of Compromise (IoCs) that can be used independently as signature-like detection artefacts for the rapid identification of known ransomware families from network traffic.

[a]e-mail: G.hatungimana@latrobe.edu.au
[b]e-mail: A.Mahmood@latrobe.edu.au
[c]e-mail: M.Chowdhury@latrobe.edu.au

Keywords Malware · Ransomware · Network Intrusion Detection System · Machine Learning

1 Introduction

Although ransomware is often considered a recent phenomenon in cybersecurity, the concept of taking computer data hostage dates back to 1989 with the creation of the AIDS Trojan. Developed by Dr Joseph Popp, an evolutionary biologist and Harvard University alumnus, the Trojan was distributed following the World Health Organisation (WHO) AIDS conference in Stockholm. Popp sent infected floppy disks by mail to conference attendees, marking one of the first known instances of malicious software designed to extort individuals for financial gain. Victims of the AIDS Trojan were initially asked to pay approximately $189 to regain access to their encrypted data [4]. Since then, new variants and ransomware families have emerged each year, evolving in both complexity, techniques and extortion techniques.

Unlike traditional computer viruses, ransomware typically does not cause permanent damage to the operating system. However, it can disable certain system services, often rendering it unusable until the victim pays the ransom. One of the most prevalent types of ransomware is crypto-ransomware, which encrypts the victim's data and demands payment in exchange for the decryption key. From a destructive point of view, crypto-ransomware remains one of the most significant threats due to its ability to lock critical data.

The strategies to combat ransomware attacks involve a combination of electronic and non-electronic methods. Non-electronic strategies primarily focus on educating users about cybersecurity threats, with a par-

ticular emphasis on social engineering techniques used by cybercriminals. Electronic strategies, on the other hand, involve the deployment of specialised software on target endpoints to prevent the successful execution of ransomware or other forms of malware. So far, electronic methods have proven to be the most reliable defence against ransomware attacks. However, the software tools currently used to detect ransomware often fall short of expectations, particularly due to the novel and advanced techniques employed by cybercriminals, which lead to Zero-Day attacks.

In addition to Zero-Day attacks, ransomware operators have increasingly adopted advanced techniques and hacking practices to compromise their targets and ensure the successful execution of ransomware payloads. This practice is commonly referred to as "evasion." To counter these evolving threats, it is recommended that detection tools be enhanced to identify incidents without relying solely on the host system's state. One potential approach is to expand the scope of detection by incorporating other sources of information, such as network traffic, which can provide additional insight into the attack.

In this paper, we propose a framework for detecting ransomware victim nodes through network traffic analysis. The framework is based on a novel network analysis technique, referred to as Traffic of Interest (ToI), and its derived Indicators of Compromise (IoC) patterns. It addresses key limitations identified in previous studies, including the lack of benign samples that exhibit ransomware-like behaviour, behavioural overlap between ransomware and legitimate activities, and limited network observability.

1.1 **Contributions**

In this paper, we propose the Network-Centric Crypto-ransomware intrusion detection framework ,leveraging network traffic to identify victim nodes. The main contributions of this research include:

1. **A Novel network traffic analysis technique is known as Traffic of Interest (ToI)**. Although numerous scholarly articles rely on session-based and time window approaches to extract patterns from multiple network packets, these approaches have inherent limitations, including the risk of missing valuable patterns, as discussed earlier or delayed detection. The ToI technique mitigates these limitations by using clear demarcation codes to define the analysis boundaries, thus reducing the potential for bias in design and the risk of missing the most valuable behavior pattern. This method is both simple and reproducible, enabling more effective comparisons of results across various studies.
2. **A new dataset.** The dataset includes traffic from 22 ransomware families spanning multiple ransomware generations, from historical to contemporary strains, and 15 benign applications deliberately selected for their behavioural similarity to ransomware. This design enables the evaluation of detection techniques against a broad range of ransomware behaviours while reducing classification bias and improving model robustness. Compared with the reviewed studies, the dataset provides broader ransomware-family coverage for binary ransomware detection and includes several families that have not been considered in previous studies as summarized in Table 1.
3. **Novel behavioral network-based IoCs**. Unlike traditional IoCs, which rely primarily on static indicators such as IP addresses, domains, and port numbers, the proposed approach focuses on behavioral characteristics extracted from network traffic. The study identifies behavioral IoCs that are consistently exhibited across multiple ransomware families and therefore remain observable even when ransomware infrastructure changes. These behavioral indicators form the basis of a continuously expandable repository that can support future ransomware detection and threat intelligence efforts.
4. **Representation of ransomware-like benign activities**. Previous ransomware detection studies commonly evaluate models using benign applications whose behavior differs substantially from ransomware activity. While such datasets are useful for benchmarking, they may not adequately represent real-world scenarios in which legitimate applications perform intensive file operations that partially overlap with ransomware behavior. To address this limitation, the proposed model is evaluated using a dataset that incorporates behaviorally similar benign applications, enabling a more rigorous assessment of its robustness and its ability to distinguish ransomware from legitimate activities exhibiting similar behavioral patterns.
   From the perspective of ransomware strain coverage, previous studies primarily considered ransomware families that emerged between 2015 and 2020. This table Furthermore, some studies evaluated their approaches using only a limited number of ransomware families, which may restrict the generalizability of the reported results to contemporary threats. In contrast, this work extends the coverage to include a broader and more diverse set of ransomware families, ranging from early families to more recent variants including. By incorporating ransomware fami-

lies that appeared between 2013 and 2022, the proposed dataset provides a more comprehensive representation of the evolving ransomware landscape and enables evaluation against both historical and recent threats.

2 Related Works

In this section, we review various techniques proposed in the literature for ransomware detection techniques that take advantage of network traffic and Machine Learning (ML). From machine learning perspectives, the detection model can be binary or multi-class. The binary detection models determine whether an instance is malicious or not, whereas multi-class models distinguish different families of malware including ransomware. Given the scope of this work, we review the recently proposed binary detection model.

As previously highlighted, machine learning-based ransomware detection using network traffic behavior has received relatively limited attention in recent research [1]. To assess the current state of the field, we reviewed studies published during the last five years using search terms such as "machine learning models for ransomware detection using network traffic". The resulting literature indicates that only a limited number of studies have explored network traffic as the primary source of features for ransomware detection, with many works instead focusing on host-based, file-system, or hybrid detection approaches.

Almashhadani et al. [5] investigated ransomware detection using network traffic analysis, focusing on the Locky ransomware family. The proposed approach extracted features from TCP, NetBIOS, and DNS traffic. For TCP communications, the authors analyzed the proportion of flows containing the RESET flag, while DNS-related features included DNS resolution errors and the entropy of domain names observed in malicious and benign traffic. A dedicated testbed was used to generate malicious and benign network traces for training and evaluation. However, the methodology used to collect and characterize the benign traffic was not described in sufficient detail. Several machine learning algorithms, including Random Forest, LibSVM, Bayes Net, and Random Tree, were evaluated, achieving detection accuracies of 99.83% with Bayes Net and 98.72% with Random Tree for packet-level and flow-level classification, respectively. Despite these promising results, the selected features exhibit several limitations that may affect their effectiveness in operational environments. First, DNS resolution errors are not unique to ransomware activity and may also result from legitimate user and application behavior, such as mistyped URLs, invalid domain names, software misconfigurations, or temporary network failures. Consequently, this feature may contribute to false positives in real-world deployments. Second, DNS resolution errors can only be observed when DNS traffic is visible, reducing their applicability in environments that employ encrypted DNS protocols such as DNS-over-HTTPS (DoH). Third, TCP RESET packets may be generated by a wide range of legitimate network conditions, including server-side connection termination, firewall policies, communication failures, and network congestion, making this feature highly dependent on the operational environment. Finally, the lack of detailed information regarding the collection and composition of the benign dataset makes it difficult to determine whether legitimate activities exhibiting behaviors similar to ransomware were adequately represented during model evaluation, potentially leading to an optimistic estimation of the false-positive rate.

Almashhadani et al [6] further explored the use of network traffic analysis for ransomware detection using the MCFP dataset. The dataset consists of 15 ransomware samples from two families, namely WannaCry and NotPetya, and five benign samples collected from Windows services. The authors constructed two distinct ransomware traffic datasets. The first dataset was session-duration based and included traffic-volume features as well as connection-level features such as external destination IP addresses and port numbers. The second dataset captured time-based communication patterns associated with SMB traffic. The session-based feature vector was used to train a session-based detection model, while the time-based feature vector was used to train a time-based detection model. The outputs of these two independent models were subsequently combined through a multiclassifier framework using a majority-voting mechanism for the final decision. The reported results achieved an accuracy of 99.8

Despite the promising results, several limitations may affect the effectiveness of this approach when deployed outside a controlled experimental environment. The first limitation relates to the selection of DNS-related features. For example, the ext-ip-addr feature relies on historical DNS queries to distinguish malicious from benign communications. However, many modern ransomware families do not rely solely on hard-coded IP addresses. Instead, they employ Domain Generation Algorithms (DGAs) to dynamically generate domain names used to locate command-and-control (C2) infrastructure, resulting in DNS query patterns that may resemble legitimate traffic. Furthermore, in enterprise environments, many legitimate applications and services communicate with external infrastructure through

Table 1: Coverage of ransomware families in related work

Family	**[5]**	**[6]**	**[7]**	**[8]**	**This Work**
Locky	*X*		*X*	*X*	
Cerber			*X*	*X*	
TeslaCrypt			*X*	*X*	
WannaCry		*X*	*X*		*X*
NotPetya		*X*		*X*	
Ryuk			*X*		
GandCrab			*X*		
Shade			*X*	*X*	
REvil/Sodinokibi			*X*		*X*
Phobos			*X*		
STOP			*X*		
RansomEXX			*X*		*X*
Maze					*X*
Ako					*X*
Coronavirus					*X*
NetWalker					*X*
AvosLocker					*X*
Hive					*X*
Ragnarok					*X*
Clop					*X*
DarkSide					*X*
CryptoLocker					*X*
BlackCat					*X*
MountLocker					*X*
Conti					*X*
Nephilim					*X*
Nemty					*X*
DoppelPaymer					*X*
LockBit 2.0					*X*
LockBit 3.0					*X*

hard-coded IP addresses. In addition, hosts often cache DNS resolutions, eliminating the need for subsequent DNS queries and thereby reducing the discriminative value of DNS-based features.

A second limitation is the method's dependency on unencrypted protocol information. The DNS and SMB features utilized by the detection models are assumed to be observable in plaintext. However, modern environments increasingly adopt encrypted protocols such as DNS over HTTPS (DoH) and SMBv3 encryption, which may conceal critical information required by the proposed feature set and consequently reduce detection performance.

The third limitation concerns the selection of benign samples. The benign traffic was obtained from an external dataset developed by the Czech Technical University (CTU). Although the dataset contains legitimate network activity, there is no assurance that it includes behaviors closely resembling ransomware operations. As a result, the trained models may not adequately learn to distinguish ransomware from legitimate activities that exhibit similar characteristics, such as large-scale file encryption, mass file modifications, file synchronization, backup operations, or bulk file deletions. Consequently, the reported false positive rate may be underestimated, potentially leading to a higher number of false positives in real-world environments.

The issue of limited observability was first addressed by the authors in [7]. They proposed a ransomware detection model for SMBv3 networks that leverages the number of TCP packets exchanged between file servers and their clients. Network traffic was categorized into three groups: packets transmitted from the client to the server (write operations), packets transmitted from the server to the client (read operations), and control commands carrying little or no payload data. The approach was evaluated using samples from twenty-six ransomware families and achieved a detection rate of approximately 99% with a Neural Network classifier. The first concern with this method is the lack of representative benign samples for establishing a realistic baseline. Benign traffic was collected from a workplace

Table 2: Summary of realated works and their limitations

Work/Author	Features / Techniques	Classification	Results	Limitations
Almashhadani et al. [5]	Session-Based TCP RESET ratio, DNS resolution errors, and domain name entropy.	Bayes Net	99.83% Accuracy	– DNS errors may result from benign user mistakes, misconfigurations, or network failures. – Encrypted DNS, such as DoH, may limit DNS traffic visibility. – TCP RESET packets may reflect legitimate network conditions. – Limited details on benign traffic composition hinder assessment of dataset representativeness and false-positive performance. – Evaluation on one (1) ransomware family may limit generalizability.
Almashhadani et al. [6]	Session-Based and Time-Based SMB, DNS and TCP.	Bagged Tree Ensemble	99.8% Accuracy, 0.0041 FPR	– DNS-based IP identification may be affected by caching, hard-coded IPs, and DGA-based ransomware. – Port and IP features may introduce dataset bias and produce false positives. – DoH and SMBv3 encryption may limit traffic visibility. – Lack of benign activities that resemble ransomware behaviour may lead to an underestimation of the performance metrics. – Evaluation on two (2) ransomware families may limit generalizability.
Berrueta et al. [7]	Session-Based SMBv3 traffic and Time-window based feature vectors counting for read, write, and control operations.	Neural Network	≈99% Detection Rate, 0.004 FPR	– Lack of benign activities that resemble ransomware behaviour may lead to an underestimation of the performance metrics. – High risk of redundant ransomware behaviour instances (57.1%)
Srinivasulu et al. [8]	DNS, HTTPS, P2P	HyQASNN	99.94% Accuracy, 99.84% Precision, 99.88% Recall	– Lack of benign activities that resemble ransomware behaviour may lead to an underestimation of the performance metrics.

environment that did not include legitimate file encryption, compression, backup operations, or large-scale file deletion activities. Consequently, the benign dataset may not adequately represent behaviours that closely resemble ransomware activity, potentially simplifying the classification task and limiting the assessment of behavioural overlap between benign and malicious traffic. A second limitation relates to the potential redundancy of ransomware instances. The study executed 77 ransomware binaries belonging to 33 ransomware families. Since variants from the same family often exhibit highly similar dynamic behaviours, many dataset in-

stances may capture nearly identical behavioural patterns. Consequently, more than 57.1% of the ransomware samples are susceptible to behavioural overlap, potentially leading to overly optimistic performance estimates.

Srinivasulu et al. [8] proposed a ransomware detection model based on a Hybrid Quantum-Attention Neural Network (HyQASNN) optimized using the Quokka Swarm Optimization Algorithm. The model was trained using samples from eleven ransomware families and benign traffic obtained from the Stratosphere IPS dataset. Examination of the original benign traffic captures indicates that the dataset primarily consists of DNS, HTTPS, peer-to-peer (P2P), and general Internet traffic. However, the authors did not disclose the complete set of features extracted from the network traces, making it difficult to assess whether the proposed approach can effectively operate on both encrypted and unencrypted traffic. The reported performance is 99.94% accuracy, 99.84% precision, 99.88% recall, and an error rate of only 0.01%. Nevertheless, the training and evaluation environments did not include legitimate file-encryption activities. Consequently, the model's ability to distinguish ransomware-induced encryption from benign encryption remains unclear and may result in elevated false-positive rates in environments where file encryption is a normal and routine operation.

Besides work-specific limitation as summarized in Table 2, generally previous studies used benign behavior baseline that do not exhibit behavioral characteristics similar to ransomware, such as file encryption, compression, and large-scale file modification. As a result, the reported performance of many proposed approaches may not accurately reflect real-world deployment conditions. Furthermore, most existing methods rely heavily on packet- or flow-level statistical features. While such features provide a useful foundation for detection, an overreliance on traffic statistics may limit model generalizability and robustness when deployed in environments that differ from the experimental setting used during training and evaluation.

As summarized in Table 1, the benign samples used for model training and evaluation are typically collected from normal workplace environments or general NIDS datasets. While such traffic is legitimate, it often lacks the intensive file access, modification, and SMB communication patterns that characterize ransomware activity. Consequently, reported detection performance may not fully reflect a model's ability to distinguish ransomware from benign applications performing similar operations. For example, Almashhadani et al. [6] evaluated their approach using Windows services and selected goodware applications, but did not provide a detailed characterization of the benign activities included in the dataset.

3 Proposed Method

In this section we discuss about the method we use for detecting ransomware intrusion. The method leverages network traffic between clients and file server. We start looking into general clients-file server operation then the proposed detection techniques.

3.1 Background

Most corporate workplace environments enforce policies and technical controls that limit the storage of sensitive data on client endpoints. One of the widely used techniques used for file sharing in the Windows environment is the Server Message Block (SMB) protocol.

SMB was developed in 1983 to enable file sharing between computers on a network, allowing remote devices (servers) to make files accessible to clients over a TCP/IP connection. Since its inception, SMB has undergone several improvements. SMBv1 was implemented with Windows Server 2003 but was found to be vulnerable to the EternalBlue exploit and was subsequently deprecated in 2003. SMBv2 replaced SMBv1 in Windows Server 2008 and Windows Vista, and is still widely used despite being susceptible to eavesdropping vulnerabilities because as all messages exchanged between the server and the client are transmitted in plain text.

The latest version, SMBv3, supports data encryption to address these security concerns. Both SMBv2 and SMBv3 are still supported and actively used. [9]. For instance, contemporary Microsoft operating systems, including Windows 10, Windows 11, and Windows Server versions such as 2019, 2022, and 2025, enable SMBv2 and later versions by default and negotiate the highest mutually supported SMB dialect during communication.

In addition to general-purpose IT systems, SMBv2 is commonly encountered in legacy and constrained environments, including Internet of Things (IoT) and Operational Technology (OT) systems, where long device lifecycles, vendor dependencies, and limited firmware update capabilities often delay adoption of newer SMB dialects. SMBv2 is also widely implemented in non-Windows platforms, such as Samba-based systems and network-attached storage devices, enabling interoperability across heterogeneous networks.

SMB data consists of commands initiated by the client to access and manipulate remote files hosted on

Table 3: Coverage of benign samples in related work

Work	**Benign Samples**
Almashhadani et al. [5]	Benign traffic generated in a dedicated testbed using Windows service traffic and goodware application. The details about which application is not reported.
Almashhadani et al. [6]	NIDS dataset developed by the Czech Technical University (CTU)
Berrueta et al. [7]	Workplace network traffic.
Srinivasulu et al. [8]	Stratosphere IPS benign traffic, including DNS, HTTPS, P2P, and general Internet traffic.

a remote server. The SMB data structure includes various headers that carry different commands [10]. As data move through the layers of the OSI model [11], they undergo an encapsulation process. As depicted in Fig.1, the SMB data is passed to the transport layer, where it is divided into smaller chunks, known as segments or streams, based on the maximum number of bytes that can be transmitted at once. The size of these segments is determined through a negotiation process governed by TCP standards. At the transport layer, the TCP header is added to each segment, which is then passed to the network layer. The network layer adds IP addresses to the segment, ensuring that they can be routed across the network. At this point, the segment is called a packet, which is forwarded to the data link layer for final delivery.

A typical use case for this protocol is an environment consisting of a File Server and multiple Client PCs. Client PCs access files hosted on the File Server via mapped network drives.

3.2 Detection Framework

From an implementation point of view, the proposed detection framework, as depicted in Fig.2, involves 2 main phases. The first phase (1&2) is signature-like, so it relies on network-based indicators of compromise for well-known ransomware families. From the experimentation, we were able to extract network packet features that correctly identify each ransomware family that we studied and we collected them into a dataset/repository. That repository can then be integrated with other network monitoring solutions/appliances for a fast track to well-known strains that could have evaded endpoint detection tools. The second phase (3&4) is machine learning-based, so it relies on features extracted from ToI-based network traffic flows. In an integrated mode, each packet entering and leaving hosts is inspected at the first phase against known ransomware IoC and raises an alarm if there is a match; otherwise, the packet proceeds to be processed at the second phase. In independent mode, one can just implement until the first phase, ignoring Machine Learning. However, he/she will not be able to detect new threats. Or, one can bypass/ignore the first phase, so all packets will be processed by ML at the second phase. In our case, we have adopted an integrated mode.

In experimentation, we discovered special packets with unique codes. We refer to such packets as packets of interest (PoI) and Indicators of compromise (IoC). We extracted codes for encrypted (SMBv3) and non-encrypted (SMBv2) network traffic. However, in order to maintain paper conciseness, we present the framework using SMBv2 codes and keep equivalent codes for SMBv3 in a separate IoC dataset.

In the second phase, we use (PoI) to collect related packets into flows referred to as Traffic of Interest (ToI). Some ransomware families also split packets into multiple chunks to avoid (IoC)-based detection. As the framework shows (2), the (ToI) technique uncovers such evasion because it normally assembles related packets. So, if an (ToI) matches (IoC) in the repository, an alarm is raised; otherwise, the (ToI) is processed by the ML module. The ML module is based on features extracted from (ToI) and standardised using the information from the file server.

3.2.1 Indicator of Compromise (IoC)

Generally, ransomware often informs the victim about the damage and provides instructions on how to make payments(**extortion**). The commonly used way to do that is by dropping extortion messages in one or more folders on the victim's device. It is the only file that does not get encrypted. The length or size of such a file depends on the information it contains; it is different from family to family. Since each family has a unique message size, it becomes an Indicator of Compromise (IoC) for well-known ransomware families. Manually, the message size can be depicted through the file properties and then stored in the IoC repository for reference during live traffic inspection. Some characteristics of such files we found so far include:

- It appears once in one folder

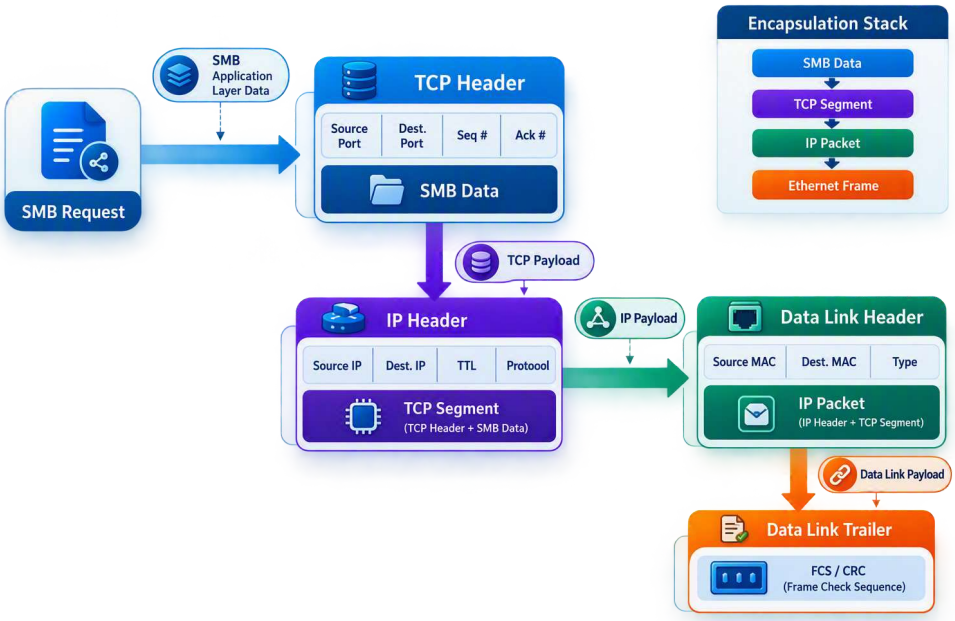


Fig. 1: SMB Encapsulation

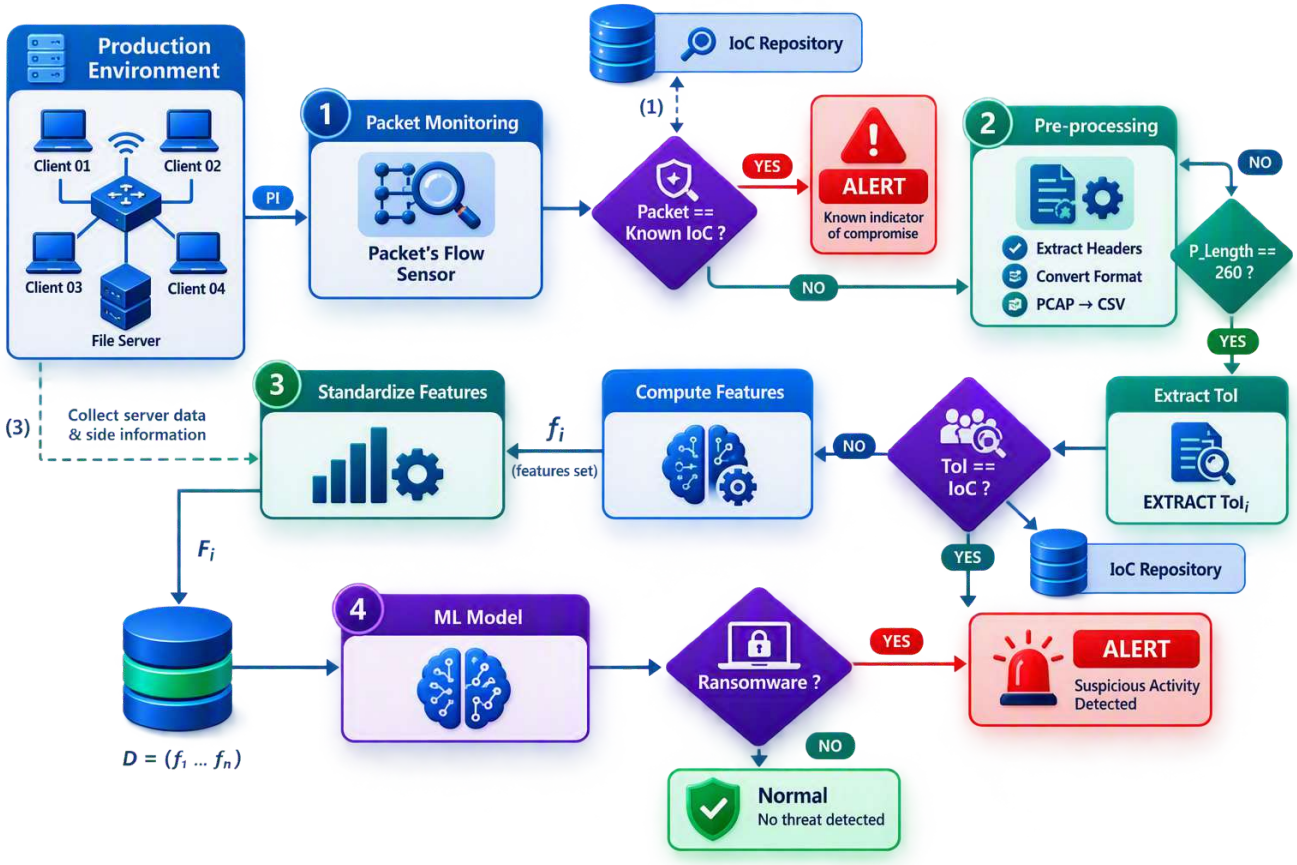


Fig. 2: Hybrid Detection Framework

- It is always a result of $C410$, a newly created file
- It is the same size across all folders
- It can be split into small chunks to evade detection

Certain variants of evasive ransomware attempt to conceal their extortion messages by fragmenting them into multiple packets to evade detection. In that case, test each ToI for evasion as per Eq. 2.

$$IoC = \frac{1}{n}\sum_{i=1}^{n} l_i, \tag{1}$$

where l_i represents the length of the i-th packet in the ToI, and $n = |ToI|$ is the total number of packets considered.

3.2.2 Traffic of Interest ToI

Processing network traffic traces for machine learning applications presents significant challenges, as individual network packets typically contain limited information on their own. Therefore, a critical step in applying machine learning to network traffic is aggregating individual packets into more informative data samples. In this research, we introduce a novel method, referred to as the Traffic of Interest (ToI) technique, to address this issue. The ToI method aims to aggregate related packets into cohesive data samples, ensuring that the aggregated information captures the necessary context for downstream analysis and feature extraction.

The Server Message Block (SMB) messages typically consist of data operation commands for session setup and termination, file opening, reading, writing, status checks, and their respective responses [10]. Our research observed that the SMB client frequently navigates through directories, enumerating contents in a back-and-forth manner. This behaviour is facilitated by a specific packet, which remains consistent in size (260 bytes) regardless of the environment or the directory paths being accessed. It serves as an indicator for an SMB client process, signalling it whether all items in the current directory have been enumerated and whether the client needs to traverse back or forward into other sub-directories. We refer to this packet as $p260$ and it is extracted using the algorithm 1.

Algorithm 1 Extracting ToI Techniques

Require: $\mathcal{P} == Network \quad Packets \quad flows$
1: Let $t_i, t_j Represent \quad two \quad consecutive \quad packet p_i and p_j$
2: Let $p260 \quad Represent \quad a \quad packet \quad with \quad length \quad == 260 \quad bytes$
3: Let $p \neq p260 \in \mathcal{P}$, where d is a network packet other than $p260$
4: $ToI_i \leftarrow \forall p \in \{t_i, j_j\}$

The procedure requires a set of network packet traces collected from the time the first $p260$ appears to the subsequent $p260$, excluding the latter. Consequently, we define the Traffic of Interest (ToI) as the set of activities between two consecutive $p260$ packets.

The encryption process takes plain text as input into the cypher algorithm in order to produce a ciphertext. That means, the original file contents are replaced with cypher text. Ransomware goes the extra mile by changing the extension of the file. We discovered that the mission is completed using special SMB commands referred to as Commands of Interest (CoI) as explained below.

1. At some point in time, a client wants to read a file that does not exist yet; So it will create it, and the server shall reply confirming that a new file has been created. This response/backward packet has been found with a static length of (410) regardless of the environment or file path. This $CoI$is referred to as $rp410$.
2. At some point in time, the client will read any file in the current directory; this forward packet has been found with a static length of (171) regardless of environment or file path. This CoI is referred to as $rq171$
3. At some point in time,, a client will write data to files in the current directory and the server shall reply with the number of bytes written to that file. This backward/response packet has been found with a static length (138) regardless of the environment or file path. This CoI is referred to as $rp138$.
4. At some point in time, the client will attempt to open a file/folder, and the server responds with $rp378$.
5. At some point in time, the client wants to set new information on files, such as renaming. This forward packet has been found with a static length of (124) regardless of the environment or file path. This CoI is referred to as $rq124$.
6. At some point in time, the client will close the open file/folder. This forward packet has been found with a static length of $rq146$.

3.2.3 Features Engineering/Extraction

Based on CoI as explained in the previous sub-section, we compute basic features (BF) using algorithm 3. For each ToI, we count the frequency of each CoI. Normally, you can save new data by creating a new file, writing on it, then saving, or by opening an existing file, writing additional data, then saving. The relevant CoI as earlier discussed are $rq171$,$rp410$ and $rp138$. $rp138$ tells us that new data has been successfully saved either following the modification on an existing file $rq171$ or by creating a new one $rp410$.

As depicted in algorithm 2, for each Write response packet, we find whether it has been triggered by reading an existing file ($rq171$) or by creating a new file ($rp410$). The latest trigger to occur before that write response is considered (the closest command to the write response). Next, from the extracted CTW and RTW, we compute total, mean, and Standard Deviation, resulting in extended features as presented in Tab.5

3.2.4 Features Standardization

The cumulative values obtained as results of algorithm 2 vary from one production environment to another, as the structure and size of the shared network directory are different and have different infrastructure setups. In order to generalise our method for different environments, we introduced a script to be installed on the file server for constantly reporting statistics of files and directories available in the shared network drive (Probing). We use the probe information; the number of files (n) of the shared drive at the time of capture to convert basic features (BF) into rates based on 100 files as per equation 1. The number of features (fields) computed and what each stands for is presented in Table 5.

$$\mathcal{SF} = \frac{\mathcal{BF} * 100}{n} \tag{2}$$

Table 4: Extracted Basic Features

Feature	**Description**	**Related** COI
Source	The origin of a packet (IP address)	n/a
Destination	The destination of a packet (IP address)	n/a
Duration	Activity duration	n/a
Read	Number of files read	171
Open	Number of files and directories opened	378
Modify	Number of newly created files	410
Write	Number of files with content modified	138
Set	Number of files with modified attributes	124
Close	Number of files and folders closed	146
Label	Ransomware family or Benign Application Name	n/a

Algorithm 2 Extended Features Computation Techniques

Require: ToI
Let $CTW = \emptyset$; Stands for Create Then Write packets as results of $p410$
Let $RTW == \emptyset$; Stands for Read Then Writes packet as results of $p171$
Let $c \in ToI = \{410, 171\}$; a set of centToIds packets
Let $rp138(i)i_{\in}\{1, ..., n\}$; any Write response packet within ToI
For each $rp138_i$:
if $\{(rp138_i - c_0) == Minimum\}$ **then**
    $CTW \leftarrow rp138_i[writecount]$
end if
if $\{(rp138_i - ci_1) == Minimum\}$ **then**
    $RTW \leftarrow C138_i[writecount]$
end if
$Letx \in RTW$
$Lety \in CTW$
$Let\mu, mean \quad of \quad x \quad or \quad y$
X $=\sum(x_i)$
$TotX = \sum_0^i(x[writecount])$
$\mu_x = \frac{TotX}{X}$
Y $=\sum(y_i)$
$TotY = \sum_0^i(y[writecount])$
$\mu_y = \frac{TotY}{Y}$
sdvX$=\sqrt{\frac{\{\sum(x_i - \mu_x)^2}{X}}$
sdvY$=\sqrt{\frac{\{\sum(y_i - \mu_y)^2}{Y}}$
$MRW_x == x[writecount]_i \in X, mostfrequent$
$MRW_y == y[writecount]_i \in Y, mostfrequent$

Algorithm 3 Basic Features Computation Procedure

Require: ToI
Let $l; a \quad apacket \quad length \quad header \quad size$
Let $P; \quad P(l) \notin \{410, 171, 138, 124, 146\}; a \quad apacket$
for each $CoI \in \{410, 171, 138, 124, 146\}$ **do**
        $BF = \forall P, \quad P(l) == CoI$
end for

3.2.5 Machine Learning

This module is used to detect novel ransomware variants by analysing Traffic of Interest (ToI) packets for anomalous behaviours that deviate from established patterns. These behaviours may include the presence of encrypted data or irregular access to shared files, which are indicative of potential ransomware activity.

Machine learning (ML) plays a pivotal role in artificial intelligence (AI) by enabling computers to mimic human thinking. ML involves various algorithms that classify data using techniques such as strata, probability, or decision trees. Each ML technique employs multiple methods to capture and manipulate patterns within existing data, allowing for the classification of unseen data. Well-known examples of these methods include Support Vector Machines (SVM), Neural Networks, Logistic Regression, Discriminant Analysis, Random Forests,

Table 5: Computed (Extended) Features

Feature	**Description**	**Related** *COI*
rtw_cnt	Number of write responses triggered by read	C171
rtw_tot	Total written data triggered by read	C171
rtw_mean	Mean of written data triggered by read	C171
rtw_stdv	Standard deviation of written data triggered by read	C171
rtw_mrw	Most repeating written data triggered by read	C171
ctw_cnt	Number of write responses triggered by create	C410
ctw_tot	Total written data triggered by create	C410
ctw_mean	Mean of written data triggered by create	C410
ctw_stdv	Standard deviation of written data triggered by create	C410
ctw_mrw	Most repeating written data triggered by create	C410

Linear Regression, Naïve Bayes, K-nearest neighbours, and Decision Trees. Notably, Neural Networks have evolved into an independent field known as Deep Learning, which has garnered significant attention in AI research. The choice of which ML technique or method to use depends on the nature and complexity of the data being analysed.

A common characteristic across all ML methods is their reliance on a benchmark dataset, which serves as the basis for training the algorithms. Once a benchmark dataset is established, the next critical step is feature selection, which involves determining which data patterns should be used to train the model. This step marks a key distinction between traditional machine learning and deep learning, particularly when dealing with large datasets. In traditional ML, human intervention is often required to select the most relevant features for training. In contrast, deep learning models, particularly neural networks, can autonomously identify which features are most relevant, minimising the need for manual feature selection.

In this work, we tested the dataset with different traditional Machine Learning techniques and evaluated the performance of each. Generally the evaluation of machine learning model is based on 3 and 4. Unlike in other domains where ML is applicable, high False negative rates expose the business to higher risk than False Positive rates. In other words, a benign application detected as malicious would only cause time waste, leaving company data intact. But a malicious application passed unseen will cause data/system damage, and financial loss and cost more time for recovery [12]. So we extends the evaluation to include false negative rate metrics as per equation 5.

$$Accuracy = \frac{(fp + tp)}{(tn + fp + tp + fn)} \tag{3}$$

$$FPR = \frac{(fp)}{(fp + tn)} \tag{4}$$

$$FNR = \frac{(fn)}{(fn + tp)} \tag{5}$$

where;
fp == number of Normal classified as Ransomware
fn== number of Ransomware classified as Normal
tp== number of Ransomware classified as ransomware
tn== number of Normal classified as Normal

4 Experimental Results

In this section, we discus the experimentation process and results. For instance we discuss how we built the dataset, extracted features, classification process and resulting results.

4.1 Dataset

E.Berrueta [13] developed and published a public data repository containing the results of executing over 70 ransomware samples collected between 2015-2020 from different families. The repository includes network traffic captures (DNS and TCP) and file Input/Output (I/O) operations generated by each sample during its activity, including the read and written bytes, the time between open operations, the number of deletions, and the file sizes. The repository comprises ransomware families collected between 2015 and 2020. While valuable for benchmarking and reproducibility, it may not capture the behaviour of more recent ransomware strains and evolving attack techniques. Therefore, the development of updated datasets remains necessary to support contemporary studies and the evaluation of modern detection approaches.

We started building a testbed where we execute samples safely. As depicted in Fig. 3, we set up a client on a 64-bit Windows 10 Professional virtual machine with specs: 8 GB of memory, and 100 GB of SSD.

The user does not have domain or local administrative rights. The user accesses the data folder through the mapped network drive. The other client PCs are for illustration purposes only and have not been used anywhere in a lab.

We set up a Windows 2016 virtual machine as a file server with specs: 8 GB of memory and 100 GB. The file server has been set up with a data folder of 200 files in 8 sub-folders. The root folder is shared with default read and write permissions, and the server is also acting as a domain controller and DNS.

The monitoring PC is a 64-bit Ubuntu 24 Virtual Machine with 8 GB of memory and 50 GB of disk. All the virtual machines run on a 64-bit Ubuntu 24 host with specs: AMD RYZEN 5000 7 processor, 32 GB memory and 500 GB SSD and are connected to a virtual switch. The port where the Monitoring VM is connected has been configured in promiscuous mode to listen to packets between Client01 and the File Server. Behind the scenes, the host is connected to the WatchGuard Firewall, which continuously logs network access activities.

4.1.1 Ransomware Sample

There are multiple publicly available sources of malware samples, including VirusShare, AnyRun, Malware Bazaar, etc. In order to download a sample, one must know the signature (usually hash values) of that malware. We searched multiple cybersecurity reports for trending ransomware attacks and related indicators of compromise (IoC) in order to get malware hashes. We realised that the most recent ransomware strains' IoCs cannot be found anywhere other than in Virustotal. Unfortunately, downloading samples from VirusTotal is not possible unless someone has a premium subscription, which costs about 600 euros monthly. So, if any IoC cannot be found in other places, it is dropped. We execute samples one by one in our testbed while capturing network traces between the client and file server for 10 minutes; a sample that does not encrypt the data folder is removed from the list (out of scope) because it might be human-operated ransomware, which requires manual operation via C2C. The final list of ransomware families covered in this work is presented in Table 7.

4.1.2 Benign Sample

In a previous work, there are a few papers with details on the acquisition of benign samples. For example, Edwardo et al [7] created benign samples by intercepting network traffic in a real production environment. Fernando et al [14] used VeraCrypt, AxCrypt, BitLocker and 7-Zip for benign.

Table 6: Ransomware Samples

Ransomware Group	**Number of ToI**
ako	8
Coronavirus	8
netwalker	41
Avoslocker	48
Hive	49
ragnarok	55
Clop	56
Darkside	58
Wannacry	62
CryptoLocker	78
alpha	79
backcat	81
mountlocker	81
Conti	87
neffilim	87
nemity	87
DoppelPaymer	88
ransomexx	92
Revil/Sodinokibi	110
lockbit2.0	121
maze	121
lockbit3.0	133

For our method, we used an extensive approach for generating benign samples. We try to mimic most of the possible normal user actions, which causes behaviour closer to that of ransomware, as explained below and presented in Table 7.

1. Data Encryption.
   Practically, in a production environment where users access centrally hosted file servers, it is rare and unsafe to allow users to encrypt their data. The IT department would rather deploy such tools on the file server; that is why most commercial encryption tools are limited to local and extended disk partitions. However, there are a few tools that users can still use to encrypt data if required.
   Hicrypt; We searched and found that **Hicrypt** is one of the few tools that can encrypt files on a network share. This tool allows a user to create an empty folder within a network share and then encrypt it. All new files which will be added to that folder later will get encrypted automatically. So we empty our network drive and encrypt it using **Hicrypt** , then we import it into the same data folder while capturing network packets.

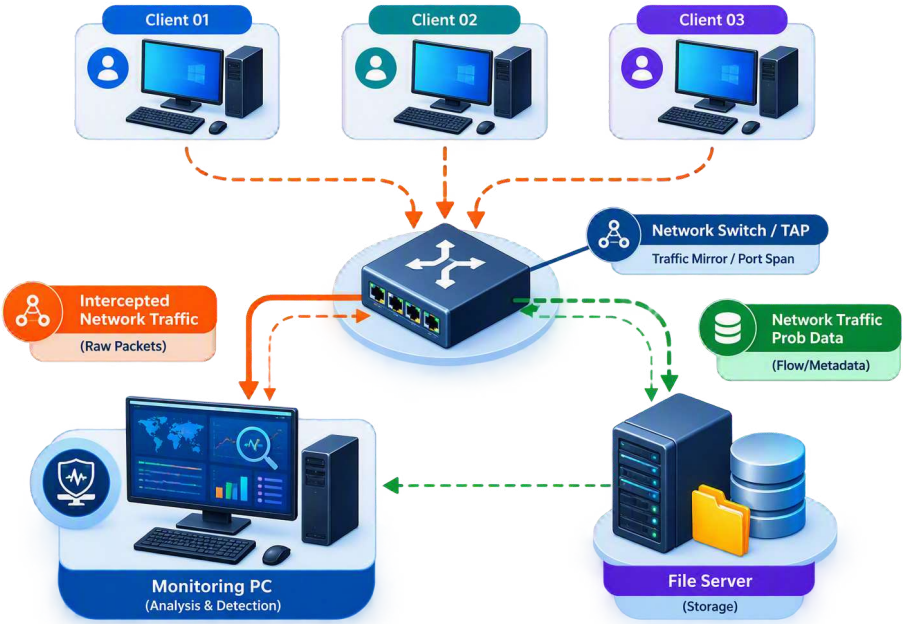


Fig. 3: Testbed Setup

Windows Encrypt Utility; Windows Encryption Utility can encrypt both local and network files. Though rarely allowed in a production environment, we decided to capture the behaviour of this action so that if it happens for some reason, our detection model should remain accurate and precise.
WinRAR, Winzip,7Zip; We realised that most file compression tools like WinRAR, WinZip and 7Zip have the option to encrypt compressed data. So we run these tools with an encryption option and then capture network traffic. The result of this operation is a compressed and encrypted file. We delete the compressed file before executing another tool.

2. Bulk Data Import and Delete.
   Among the actions that users generally perform on a network share are importing a bunch of data from other storage. This action triggers behaviour closer to ransomware due to the fact that many new files get created, and a lot of data is written at the same time. To mimic this action, we used **Windows copy utility** as the most used tool in real life. We also used third-party tools **TeraCopy** and **UltraCopier** because some users prefer them over Windows utilities. We empty the network drive, then paste it in the same folder being used for ransomware analysis and the other way around.
3. Data compression.
   Sometimes, a user might need to transfer large amounts of files via email, ftp, for sharing with external partners. This can be achieved by compressing multiple files and folders into one. By doing that, a new file (container) is created, and files change extensions pretty much like what ransomware does when it changes the extension of all files after encryption and re-creates file contents. While there are many third-party tools for compressing files, we chose to use **WinRAR, WinZip, and 7-Zip** because they are the most used tools in the Windows environment. WinRAR and WinZip have output extensions that a user can choose from (.zip and .rar). We run these tools while capturing network traffic.

4.2 Features

By default, Wireshark displays a limited set of fields for captured network traces, as shown in the Default Group Tab.8. To enhance the dataset for our experiment, it is necessary to include additional information beyond the default fields. To achieve this, we configure Wireshark to incorporate the extra fields listed in the Additional Group Tab.8. Once the necessary fields are configured, we export the network traces in CSV format. This format is selected for its easy manipulation and efficient handling of the data during the subsequent stages of analysis and processing.

Since SMB commands are transmitted over TCP connections, duplicate packets can occasionally appear due to the connection-oriented nature of TCP. To address this, the module is configured to identify and re-

move duplicates. Some of the additional fields included in the preprocessing are specifically aimed at supporting this task, as the default fields alone may inadvertently lead to the removal of important packets.

The final output of this pre-processing module is a clean CSV file, free from duplicates and containing all the necessary information required for further analysis in the subsequent steps.

4.3 Classification

In order to train a Machine Learning model, use the dataset that we created as per the previous sub-sections. The original dataset from the testbed is imbalanced with 844 benign and 1600 ransomware data samples.

We removed data samples corresponding to IoC. We do this because these data have well-known and distinct characteristics for all ransomware families that cannot be found anywhere with benign samples. These data are collected in a separate repository of IoC as previously explained to serve for IoC detection phases. This results in reduced ransomware data samples up to 1262. As it is always a good idea to use a balanced dataset when training an ML model, we used a Python function to select random 844 from 1262 ransomware samples. This task was done while the ransomware samples still had labels (Ransomware Family Name) to make sure that all families are represented. Then we labelled instances of the resulting sub-dataset as ransomware.

Supervised Machine Learning, required labelled sample data for training and unlabeled ones for testing. We split the whole dataset into training and testing sub-datasets or by using K-fold cross-validation [15]. Splitting a dataset is most of the time achieved by random sampling. The problem with the method is that a single run may select irrelevant data that influences biased learning. K-fold cross-validation techniques, on the other hand, split the dataset into multiple sample groups (K), resulting in a less biased learning model. So in this research, we used 10-fold cross-validation with different.

Table 7: Benign Samples

Tool	**Action**	**Number of ToIs**
Windows Utility	Delete	11
	Copy-In	4
	Copy-Out	8
	Encrypt	50
WinRAR	Zip (.zip)	1
	Zip (.rar)	11
	Encrypt	4
WinZip	Zip (.zip)	8
	Encrypt	652
7-Zip	Zip (.zip)	8
	Zip (.7z)	4
	Encrypt	1
TeraCopy	Copy	43
UltraCopier	Copy	69
HyCrypt	Encrypt	1

Table 8: Required SMB Headers

Category	**Field**	**Description**
Defaults	No	The index number of the packet
Defaults	Time	Time (seconds) elapsed since the capture started
Defaults	Source	Where the packet is coming from
Defaults	Destination	Where the packet is going
Defaults	Protocol	The type of protocol used for this communication
Defaults	Length	The length (in bytes) of the packet
Defaults	Info	The information carried in the packet (payload)
Additional	smb2-header-len	SMB header length
Additional	smb2-setinfo-size	The size of data contained in SMB Set Command
Additional	smb2-writecount	The size of data contained in SMB Write Command

Normal Detection (Not Early) .

With normal detection, we refer to detection without caring about the amount of files damaged before a detection occurs. We tested the dataset with different machine Learning Models. As the results show in the Table 9, Random Committee classification exhibits the best performance with 99.6% accuracy, 0% False Negative Rate (FNR) and 0. 004% False Positive Rate (FPR). This result represents the best performance achieved in our experiments, demonstrating both robustness and high efficacy.

Table 9: Performance Metrics of Various Classifiers

Classification	**Accuracy**	**FPR**	**FNR**
RandomCommitee	99.645	0.004	0.001
RF	99.585	0.004	0.001
Rules.Part	98.874	0.011	0.010
lazy.ibk	98.519	0.015	0.015
BayesNet	98.045	0.020	0.016
Logistics	97.275	0.027	0.022

For comparison purposes, we also create models for only 1 (Lockbit2.0) and two (Lockbit2.0 & Maze) ransomware families, as has been done in [6]. In both scenarios, the accuracy achieved is 100 % accuracy. However, we caution that such models are not robust due to the limited scope of ransomware families considered. Given the large and evolving number of ransomware families, we do not recommend this approach for broader applications.

Early Detection .

Early detection refers to a kind of model in which dynamic features are captured before the executed payload completes all tasks. The objective is to detect intrusion before it causes the most damage [16]. Different scholarly articles, such as [17], proposed methods exclusive to ransomware early detection on a host-based basis. Berrueta et al.[18] proposed a method using a threshold for early detection of ransomware based on SMB network traffic.

In our dataset, the smallest ransomware data sample size is 8 for Ako and Coronavirus. So, for each ransomware family, we first created a sliding window of 8 ToI. That means that the first test scenario has 176($8x22$), the second has 352($16x22$), etc. For each test scenario, we randomly selected an equivalent benign sample from a benign dataset and then classified the data in the same way as for full detection. The window size that produces high accuracy is considered the best for early detection. As shown in Fig. 4a, the first 48 ToI produced 99.44 % accuracy, just 0.24 % away from full detection accuracy.

Since different ransomware families have different numbers of ToI, it is obvious that the degree to which each of the families can be detected at an early stage is different. For example, Fig. 4b shows that Lockbit can be fully detected with 36 % of its execution. Only Ako, coronavirus, netwalker and avoslocker require completion execution as they produced quite a small number of ToIs.

5 Conclusion, Discussion & Future Work

In this paper, we present a framework for crypto ransomware intrusion detection based on network traffic analysis. The proposed framework is founded on a novel network traffic analysis concept referred to as *Traffic of Interest* (ToI) and comprises two main modules: an Indicator of Compromise (IoC) module and a Machine Learning module. The IoC module operates in a signature-based mode, inspecting both individual and aggregated network packets against known indicators of compromise, while the machine learning module is responsible for identifying previously unknown threats. The results presented in this paper focus on the machine learning component of the framework.

Using the Random Committee classifier, the proposed model achieved an accuracy of 99.645%, a false positive rate (FPR) of 0.004, and a false negative rate (FNR) of 0.001. Although some related studies reported slightly higher accuracy values, direct comparison of performance metrics should be interpreted with caution due to differences in ransomware-family coverage, benign sample selection, feature extraction techniques, and evaluation methodologies. In particular, the proposed dataset incorporates benign applications selected for their behavioural similarity to ransomware-related activities and includes several contemporary ransomware families that were not considered in most of the reviewed studies.

The experiments were conducted using both encrypted and unencrypted SMBv3 network traffic. The $IoCs$ presented in this paper correspond to unencrypted traffic; however, equivalent indicators were identified for encrypted traffic. The study includes 22 ransomware families and 15 benign applications. All ransomware and benign samples were collected and executed within the same testbed environment to ensure consistency during data collection. The benign applications were deliberately selected to perform activities commonly observed in enterprise environments and that may exhibit behavioural characteristics similar to ransomware operations.

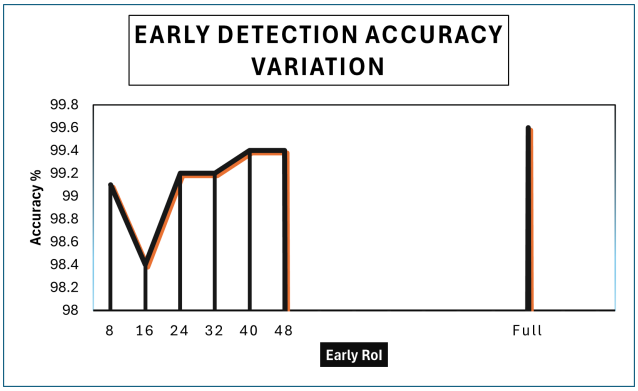
Through this analysis, we identified and extracted a set of novel network-based $IoCs$, which were subsequently stored in a dedicated repository for use by signature-based and rule-based detection systems. Future work will focus on expanding the dataset with additional ransomware families, continuously updating the repository of behavioural indicators, and evaluating the framework across a broader range of operational environments.

Funding

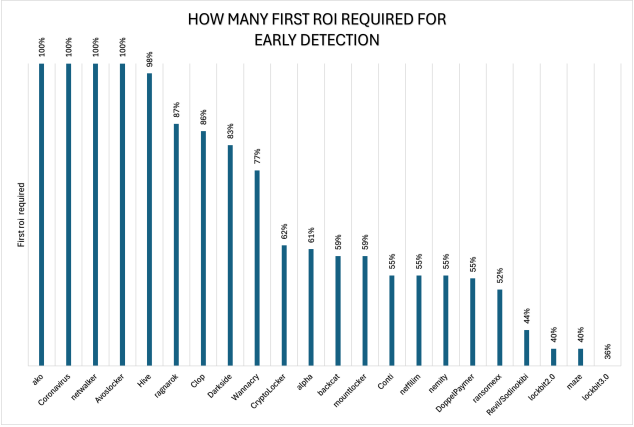
This research did not receive funding.

References

1. A. Alshamrani, M. Alghamdi, IEEE Access **12**, 1 (2024). DOI DOI:10.1109/ACCESS.2024.3397921
2. K. Sridevi, W. Masood, S. Tanveer, O.M. Mirza, N.R. Sivakumar, S. Selvarajan, Scientific Reports **16** (2026). DOI 10.1038/



(a) Early Detection Accuracy



(b) Early Detectability

Fig. 4: Comparison of early detection performance metrics

s41598-026-40871-2. URL https://doi.org/10.1038/s41598-026-40871-2

3. Q. Li, K. Xu, IEEE Communications Surveys & Tutorials **25**(1), 791 (2023). DOI 10.1109/COMST.2022.3208681
4. R. Smith, Computer Security Journal **6**(1), 58 (1990)
5. A.O. Almashhadani, M. Kaiiali, S. Sezer, P. O'Kane, IEEE Access **7**, 47053 (2019). DOI 10.1109/ACCESS.2019.2907485
6. A.O. Almashhadani, D. Carlin, M. Kaiiali, S. Sezer, Computers and Security **121** (2022). DOI 10.1016/j.cose.2022.102860
7. EduardoBerrueta, D. Morato, E. Magaña, M. Izal, Expert Systems with Applications **209** (2022). DOI 10.1016/j.eswa.2022.118299
8. M. Srinivasulu, M. Prasad, M. Elangovan, P.S. Geetha, Neurocomputing **671**, 132524 (2026). DOI 10.1016/j.neucom.2025.132524. URL https://doi.org/10.1016/j.neucom.2025.132524
9. PacketSafari. Smbv1 vs smbv2 vs smbv3: Understanding the differences and securing your network (2020). URL https://www.packetsafari.com/blog/2020/06/09/smbv1-vs-smbv2-vs-smbv3/. Accessed: 9 September 2025
10. W. Foundation. Smb2/write - wireshark wiki (n.d.). URL https://wiki.wireshark.org/SMB2/Write. Accessed: 9 September 2025
11. O. Corporation. Data encapsulation and the tcp/ip protocol stack (system administration guide, volume 3) (n.d.). URL https://docs.oracle.com/cd/E19455-01/806-0916/ipov-32/index.html. Accessed: 7 February 2025
12. P. Maniriho, A.N. Mahmood, M.J.M. Chowdhury. A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges (2022). DOI 10.1016/j.future.2021.11.030
13. E. Berrueta, D. Morató, E. Magaña, M. Izal, IEEE Access **8**, 65658 (2020). DOI 10.1109/ACCESS.2020.2984187
14. D.W. Fernando, N. Komninos, Computers and Security **137** (2024). DOI 10.1016/j.cose.2023.103629
15. J. Brownlee. k-fold cross-validation (2020). URL https://machinelearningmastery.com/k-fold-cross-validation/. Accessed: 9 September 2025
16. M. Cen, F. Jiang, X. Qin, Q. Jiang, R. Doss. Ransomware early detection: A survey (2024). DOI 10.1016/j.comnet.2023.110138
17. X.Z. Deng, M.C. Cen, M. Jiang, M. Lu, Cluster Computing (2023). DOI 10.1007/s10586-023-04043-5
18. D. Morato, E. Berrueta, E. Magaña, M. Izal, Journal of Network and Computer Applications **124**, 14 (2018). DOI 10.1016/j.jnca.2018.09.013