Scientific Research Publishing

Journal of Information Security, 2023, 14, 264-293
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Majority Voting Ransomware Detection System

Simon R. Davies*, Richard Macfarlane, William J. Buchanan

School of Computing, Edinburgh Napier University, Edinburgh, UK
Email: *s.davies@napier.ac.uk

How to cite this paper: Davies, S.R., Macfarlane, R. and Buchanan, W.J. (2023) Majority Voting Ransomware Detection System. *Journal of Information Security*, **14**, 264-293.
https://doi.org/10.4236/jis.2023.144016

Received: April 21, 2023
Accepted: August 13, 2023
Published: August 16, 2023

Abstract

Crypto-ransomware remains a significant threat to governments and companies alike, with high-profile cyber security incidents regularly making headlines. Many different detection systems have been proposed as solutions to the ever-changing dynamic landscape of ransomware detection. In the majority of cases, these described systems propose a method based on the result of a single test performed on either the executable code, the process under investigation, its behaviour, or its output. In a small subset of ransomware detection systems, the concept of a scorecard is employed where multiple tests are performed on various aspects of a process under investigation and their results are then analysed using machine learning. The purpose of this paper is to propose a new majority voting approach to ransomware detection by developing a method that uses a cumulative score derived from discrete tests based on calculations using algorithmic rather than heuristic techniques. The paper describes 23 candidate tests, as well as 9 Windows API tests which are validated to determine both their accuracy and viability for use within a ransomware detection system. Using a cumulative score calculation approach to ransomware detection has several benefits, such as the immunity to the occasional inaccuracy of individual tests when making its final classification. The system can also leverage multiple tests that can be both comprehensive and complimentary in an attempt to achieve a broader, deeper, and more robust analysis of the program under investigation. Additionally, the use of multiple collaborative tests also significantly hinders ransomware from masking or modifying its behaviour in an attempt to bypass detection. The results achieved by this research demonstrate that many of the proposed tests achieved a high degree of accuracy in differentiating between benign and malicious targets and suggestions are offered as to how these tests, and combinations of tests, could be adapted to further improve the detection accuracy.

Keywords

Ransomware Detection, Malice Score, Score Card, Malware, NapierOne Dataset

1. Introduction

Crypto-ransomware infections remain a significant threat to governments and companies alike with high-profile cyber security incidents regularly making headlines [1] [2]. The detection of ransomware is often described as an arms race [3] between threat actors and the people responsible for developing effective malware countermeasures and techniques.

There are two main approaches used in malware analysis in general and ransomware analysis in particular Static Analysis [4], where the evaluation of the program is performed without the actual execution of the code. Essentially the program contents are examined in an attempt to determine the nature of the program and its possible application. This is normally achieved by attempting to isolate and identify known patterns or signatures within the code. Static analysis scales well and can provide better coverage of a ransomware binary code. However, static analysis can produce false execution behaviour as code paths may not be reachable during actual execution [5] and tell-tale signatures may not be known at the time of analysis. Dynamic Analysis, on the other hand, executes the program under investigation in an instrumented or monitored manner and garners more factual information on the behaviour and effect of the program. Dynamic analysis can provide more accurate information on the actual execution behaviour of the investigated binary, though dynamic analysis can be computationally expensive [6] and contains some element of risk.

The problem of automatic malware detection is a difficult one, with no full solution in sight despite decades of research [7]. The traditional approach, based on analysis of static signatures, is increasingly rendered ineffective by polymorphism and the widespread availability of program obfuscation tools [8] [9]. Using such tools, malware creators can quickly generate thousands of binary variants of functionally identical samples, effectively circumventing signature-based approaches. As a result, in recent years, the focus of the research community has increasingly shifted toward dynamic, behaviour-based analysis techniques. Behavioural approaches sidestep the challenges of obfuscated binary analysis. Instead, they focus on the run-time behaviour of malware processes, which is difficult to alter without breaking core functionality and is therefore considered a reliable fingerprint for malware presence [7].

Over the years many different detection systems have been proposed as solutions to the ever-changing dynamic landscape of ransomware detection. These approaches have leveraged many different techniques such as machine learning [4] [10] [11] [12] [13] [14], neural networks [15] [16] [17] [18], file entropy [14] [19] [20] [21] [22], kernel hooking and process behaviour [23] [24] [25] [26]. In the majority of cases, the described systems propose a method based on the result of a single test performed on either the executable code, the process under investigation, its behaviour or its output. Many of the proposed systems claim to archive relatively high accuracy. Unfortunately, the researchers rarely publish enough detail of their research or the datasets used to allow the reported results

to be replicated. Berrueta [27] identifies that there are no common metrics of accuracy and performance in ransomware detection. The fragmentation of scientific research on ransomware combined with a lack of coherent investigation methodology is a major challenge in this research [28]. This view is supported by Maigida [29] who states that the lack of readily available data is also hindering the speedy development of detection and prevention solutions.

In a small subset of ransomware detection systems, the concept of a scorecard is employed. In these specific detection systems, multiple tests are performed on various aspects of a process under investigation. The results of each test contribute to an overall score for the process. A decision can then be made, based on this score, as to whether the process under investigation is benign or malicious. The main proponent of this approach was Kharraz [30] in their implementation of the *Redemption* detection system. In this work, they refer to this cumulative score as a *Malice Score*, and for the remainder of this paper, we will use their terminology when discussing this combined ranking score. Other detection systems that have also used this concept of a cumulative malice score are [31] [32] [33] [34] [35].

None of the described systems used an analytical or algorithmic approach to calculating values that could then be combined into a cumulative malice score, rather they relied on some form of machine learning to determine the result. This paper describes the work performed by the authors in building on the original research conducted by Kharraz [30], enhancing and updating their approach and proposing many new discretely calculated static and dynamic analysis tests that could be incorporated into the final malice score calculation.

A majority voting approach was chosen for the ransomware detection system proposed in this work. With this type of system, each of the underlying contributing tests generates a binary output. The result of an individual test can be either that it is considered malicious or it can be considered benign. These individual contributing scores are calculated using algorithmic rather than the heuristic techniques previously proposed in earlier research. Once all the tests have been performed, the resulting votes are then collated into two sets, malicious votes and benign votes. The final classification decision of the detection system is then determined from the set that received the majority of votes. An advantage of this approach is that the system requires no training, as the constituent values are calculated using discrete reproducible tests that require no prior knowledge or model training. These proposed new additional tests are validated using a modern and diverse dataset [36] to determine both their accuracy and viability for use within a ransomware detection system. In the initial design, each test has an equal weighting and thus an equal contribution to the final result. However, this design may be adapted in later iterations by the inclusion of weighting and bias to the results of individual tests, allowing their votes to have more effect on the final decision.

There are many benefits associated with using a cumulative score calculation approach to ransomware detection. For example, when using such an approach,

the detection system does not rely on a single specific attribute to base its decision on whether the program under investigation is malicious or not. Rather it can leverage multiple tests that can be both comprehensive and complimentary in an attempt to achieve a broader, deeper and more robust analysis of the program under investigation. Also, such a system would be easier to enhance, as adding additional tests based on new research would be straightforward. Bias from one particular test [4] [7] would also be mitigated, and the weighting of each contributing test could be adjusted to improve accuracy. Additionally, the use of multiple collaborative tests also significantly hinders ransomware from masking or modifying its behaviour in its attempt to bypass detection [7] [37].

The remainder of the paper is structured as follows. In Section 2, we discuss some of the main techniques used in ransomware detection and discuss in detail other techniques that use a collaborative voting approach or a combined scoring technique. In Section 3, we provide a description of the candidate tests that could potentially be included in the cumulative malice scoring calculation and outline the methodology used in the experiments. In Section 4, we present the recorded results and discuss the consequences of the findings with regard to the development of anti-ransomware techniques, and we provide some recommendations for crypto-ransomware detection approaches moving forward. Finally, in Section 5, we discuss the main findings and conclusions gained from this research together with possible limitations in using this approach and suggest further research that could be conducted based on the findings from the research presented in this paper.

The main contributions of this paper are:

- Design, development and detailed description of 23 potential ransomware detection tests.
- Investigation into the amount and frequency of Windows API calls within the ransomware executable files and volatile memory of a ransomware process.
- Validation of the effectiveness of the proposed tests in detecting ransomware.
- A ransomware detection system based on algorithmic derived ransomware indicators.
- The use of a modern publicly available dataset during the development and testing of the system. The majority of the similar systems proposed in the literature use datasets that are up to 14 years old.

2. Related Work

Over the last 20 years, a significant number of ransomware detection systems have been proposed in the research literature. The approaches used by these detection systems can be loosely divided into two categories. In one approach, a single method or test is developed which is then used to determine if the system is being attacked by ransomware. The alternative approach is to use machine learning to perform the identification. With the machine learning approach, the system designers identify key features from the running process and system un-

der investigation. The machine learning model then attempts to determine patterns within these features on which to base its judgement. A decision, or classification, is then made, based on the measured values of these features, as to whether the system is under attack or not.

Examples of single-method approaches are [10] [12] [38] [39] [40] [41]. In these cases, the entire effectiveness of the detection technique relies solely on the ability of this single criterion to distinguish between benign and malicious programs [42]. For example, one particular technique used in the identification of ransomware execution is to use the calculated entropy value of the files created by a process. Encrypted files tend to have a high entropy value whereas the entropy value of plain text files is much lower. Encrypted output files generated during the execution of a ransomware program would tend to have higher entropy values, possibly allowing them to be identified as a product of a ransomware infection. Unfortunately, this technique struggles to correctly distinguish between encrypted files and benign files that also have high entropy such as compressed files. The use of entropy as a detection metric has also been called into question [37] [43] as there exist techniques that could be used by ransomware to avoid detection via encoding or, in some other way, manipulating the encrypted output file.

Examples of ransomware detection techniques that have leveraged machine learning are [4] [10] [11] [12] [13] [14] or similarly neural networks [15] [16] [17] [18]. These systems are trained using extracted features from typical ransomware processes or systems that are being attacked by ransomware. Examples of features that are used in these systems are: write entropy, file overwrite behaviour, directory traversal, directory listing, cross-file type access, read/write/create/close operations, temporary files, file type coverage, file similarity, file type change and access frequency [42]. In most cases, with systems that rely on machine learning to determine if a system is being attacked, the significance of the individual extracted features and their subsequent impact on the final classification is represented internally by the detection system's model and is not immediately obvious to an observer. Inadequacies with this approach have been investigated in the literature [42] which discusses classifier evasion techniques, known as adversarial machine learning that can be leveraged by ransomware developers to avoid classification and subsequent detection.

However, in a few proposed ransomware detection systems, the designers do try to provide insight into the machine-learning techniques used and how the tested features affect the overall decision-making process. The developers of the detection system UNVEIL [34] and its successor Redemption [30], introduce the concept of a *malice score* which is a combined weighted score derived from the outcome of individual feature tests. The system detects suspicious activity using dynamic analysis and generates a malice score using a heuristic function. Inputs to this function are various behavioural features such as file entropy changes, writes that cover extended portions of a file, file deletion, processes writing to a large number of user files, processes writing to files of different types and back-

to-back writes. CryptoLock [44] propose a similar approach summing the results of various tests into a cumulative scoring they refer to as a *Reputation Score*. This score is derived from measurements of file type changes, the similarity between original and written content and output file entropy values. Another detection system, RWGuard [35], does mention the specific features that are inspected and include file IO, decoy files, file change monitoring and crypto API monitoring. However, very little detail on how the specific calculations are performed is provided. DNA-Droid [45], was the only detection system found that, leveraged a combination of static and dynamic analysis as the inputs to their neural network model. In all cases if this cumulative score is above a certain threshold, then the process is deemed to be malicious, otherwise, the process is considered benign.

However, in all these cases, the individual test results and thresholds are still determined heuristically via the machine-learning model. The model itself decides the significance and weighting given to each extracted feature and the influence that each feature has on the final classification. Reducing the entire decision-making process to effectively a black box function. A consequence of this is that it is difficult for the designers to directly affect the final decision, thus preventing them from being easily able to tune and influence the decision-making process and final classification produced by the model. The resulting quality and accuracy of the decisions made by these systems are essentially reliant on the quality of the training data used to develop the models in the first place.

No ransomware detection systems have been identified in the literature that uses a malice scoring type approach where the constituent scores contributing to the final malice score are determined using analytical or algorithmic calculation methods as opposed to the heuristics used in machine learning approaches.

3. Methodology

This section introduces a collection of potential tests that could be used in collaboration to determine if a process is malicious or benign. There is a binary outcome for each of these tests with a test failure indicating that the subject of the test is more likely to be malicious and passing the test indicating that it is more likely to be benign. The resulting votes from each test are then recorded. Each of these proposed test results would then contribute to the final overall *malice score* of the process under investigation. Each contributing test has the same weighting and thus the same impact on the final scoring. After all the tests have been conducted the classification decision is made, based on an aggregation of received votes, malicious or benign. A conceptual overview of how the proposed system would be configured is shown in **Figure 1**.

3.1. File Content Analysis

This collection of tests is performed on any output produced by the process. In the majority of cases, this would manifest itself as files being written to disk. This

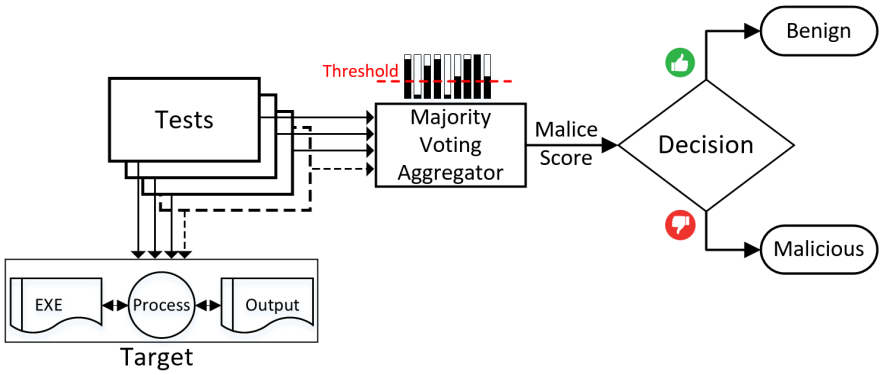

Figure 1. Overview of proposed system.

behaviour is common for processes such as editors, web downloads, email clients, system logging, compression programs, as well as the output from crypto-ransomware programs. These tests will use both the content of the file being written as well as metrics derived from the file's metadata such as file name and extension.

The NapierOne [46] data set was leveraged in many of the tests that rely on file analysis. This data set is ideally suited for this task as it contains many examples of the most commonly used file types. The data set contains 5000 example files for each of the prevalent file types shown in **Table 1**.

Apart from the normal file types found in typical use, the NapierOne data set also contains example files that have been encrypted by the ransomware strains shown in **Table 2**. The data set contains 5000 example encrypted files for each of these ransomware strains. (The SHA256 hash values for these ransomware strains are provided in **Table B1** which appears in the **Appendix**). According to previous work [14] [47] the use of diverse families of ransomware strains is more important than the number of ransomware samples from a few families for evaluating the performance of ransomware detectors. It is because the core behavioural traits shown by crypto-ransomware in encrypting data attack do not change from one variant to the other within a family [14].

The entire dataset used during this research contains 365,000 files covering 73 separate and distinct file types and is publicly accessible at www.napierone.com. The dataset contains 210,000 benign files from the 42 different file types shown in **Table 1** and 155,000 encrypted files from the 31 ransomware strains shown in **Table 2**.

File Magic Number Test. Magic numbers are usually the first few bytes of a file. These are normally unique to a file format and can be used to identify many common types of files [48]. While not all files contain this signature, for example, plain text files such as CSS, CSV, JSON, SVG, TXT and XLST, file types such as DOCX, PDF, XLSX and many others do contain this unique value. An extensive search was performed in an attempt to generate a comprehensive list of commonly used file types [36] [48]-[53] and where possible the corresponding magic number and typical file extension for that type. This research resulted in

Table 1. NapierOne file types.

Type	Type	Type	Type
7ZIP	EPS	MP3	SVG
APK	EPUB	MP3	RAR
BIN	EXE	MP4	TIF
BMP	GIF	ODS	TXT
CSS	GZIP	OXPS	WEBP
CSV	HTML	PDF	XLS
DLL	ICS	PNG	XLSX
DOC	JS	PS	XML
DOCX	JPG	PPT	ZIP
DWG	JSON	PPTX	
ELF	MKV	RAND	

Table 2. NapierOne ransomware strains.

Strain	Strain	Strain
AVOSLOCKER	DARKSIDE	PHOBOS
BADRABBIT	DHARMA	RAGNAR
BLACKBASTA	GANDCRAB	RANSOMEX
BLACKCAT	HELLOKITTY	RYUK
BLACKMATTER	JIGSAW	SODINOKIBI
CERBER	LOCKBIT	SUNCRYPT
CHIMERA	LORENZ	TESLACRYPT
CLOP	MAZE	WANNACRY
CONTI	MEDUSALOCKER	WASTEDLOCKER
CRYPTOLOCKER	NETWALKER	
CUBA	NOTPETYA	

the creation of a reference list of more than 600 entries of documented magic numbers and corresponding file extensions.

This test focuses on determining the magic number of the file under investigation and then comparing it with the file name's extension to confirm that they correlate. As plain text files do not have a magic number, then these were excluded from this test. The test was then applied to all the remaining files within the test dataset. For a file under test, if its magic number matched the corresponding expected file extension, the test passed and the file was considered benign, otherwise, the test failed and the file was considered a possible consequence of malicious activity.

Printable Characters Test. This is a complimentary test and is only run on files that do not usually contain a magic number. As these are plain text files,

then the majority of their contents should contain printable ASCII characters. Examples of files of this type are markup files such as HTML or plain text documents such as TXT. The definition of printable characters are characters that have an ASCII value between 32 and 126 as well as the format control characters which have ASCII values between nine and 13. From analysing the nearly 50,000 plain text files in the NapierOne dataset, it was found that on average plain text files contain at least 98% printable ASCII content.

The test was then applied to all the plain text files within the test dataset. For a file under test, if its printable ASCII content was above 98%, then the test passed and the file was considered benign, otherwise, the test failed and the file was considered a product of malicious activity.

File Entropy Test. A reoccurring theme within many crypto-ransomware detection techniques is the concept of randomness and file entropy. Researchers assert that a good indicator [37] [54] [55] [56] of crypto-ransomware activity is the generation of files whose contents appears to be random and contain no distinguishable structure. It is agreed that Well-encrypted data should be indistinguishable from random data [57]. Traditionally researchers in crypto-ransomware detection have chosen to use the value known as Shannon entropy [58] when calculating this metric, however, in this research, it was decided to use the chi-square [59] method of calculating this metric based on the findings of Davies [60].

The test was then applied to all the files within the test dataset. For a file under test, if its Chi-Square entropy probability value was less than 0.01 [61], then the test passed and the file was considered benign. Otherwise, the test failed and the file was considered the product of malicious activity.

BitByte Value Test. This test is based on the method described by Davies [62] which successfully distinguished between encrypted files and all other file types. This method is particularly effective at differentiating between encrypted and compressed files. A separation which previously has been proven in the past to be problematic to achieve with a reasonable level of accuracy. Essentially this test is performed by profiling the entropy distribution of the first few hundred bytes of the file under examination and comparing this profile with the entropy distribution of a control file. The difference in entropy profiles is then calculated and a value known as a *BitByte* value is determined. Files that produce lower BitByte values have a higher probability that their contents are encrypted. The research [62] identified that any BitByte value below 56, indicates with high probability, that the file is encrypted and thus possibly a consequence of a ransomware infection.

The test was then applied to all the files within the test dataset. For a file under test, if its BitByte value was greater than 56, then the test passed and the file was considered benign. Otherwise, the test failed and the file was considered a product of malicious activity.

Ransom Note Creation Test. During a crypto ransomware attack, one action often performed by the malicious process is to generate a *Ransom note* file. The

purpose of this file generation is two-fold. Firstly, to inform the user that their files have been encrypted and that they are the victim of a ransomware attack. Secondly, the file's contents will usually provide the victim with instructions on how they can recover from the attack and retrieve their files. The *Ransom note* normally explains how the victim should transfer a specific amount of crypto-currency to the perpetrator of the attack in exchange for help in recovering the affected files. There are normally several characteristics of this *Ransom note* file that can be used to distinguish it from other files. The file is normally below one KB in size, is plain text and usually contains some specific keywords such as: *encrypted*, *ransom*, *tor*, *onion*, *recover*, *wallet*, *bitcoin* [63]. In this test, the actual file name is also analysed for typical ransom note file name strings such as:*decrypt*, *readme*, *restore and helpme*. It has been identified that often these ransom note files are created prior to the actual encryption of the target files, so the identification of the creation of ransom notes would thus prove to be a good predictor of impending file encryption. This approach was leveraged in the HelDroid [64] ransomware detection system and utilised a text classifier that applies linguistic features to detect threatening text.

The test was then applied to all the files within the test dataset. For a file under test, if it is of limited size and its contents contain one or more of the trigger keywords, then the test failed and the file is considered malicious. Otherwise, the test passed and the file was considered benign.

3.2. File Name Analysis

This collection of tests is performed on the actual string value of the name of the file being written. It has been a well-known phenomenon from crypto-ransomware attacks that as well as encrypting the file contents, in the majority of cases, the affected file names will also be modified. For example by adding an extra extension or changing the original file's name. This set of tests focuses on attempting to identify this change and will again leverage the content of the NapierOne data set.

File Name Entropy Test. This test calculates the Shannon [58] entropy value of the entire file name including any extensions that it may have. In normal operation, users tend to use lower entropy strings when naming their files. An analysis of the original file names used to populate the NapierOne dataset shows that the average Shannon entropy of a file name is below six bits. This calculated value proves to be also language-independent [65]. In many cases, when ransomware alters the contents of a file, it also alters the name of the file. Common ransomware file name manipulations are the addition of random strings to the name or its extension. This action would then increase the entropy of the affected file's name.

The test was then applied to all the files within the test dataset, using their original file names. With regards to the files generated from the execution of ransomware, then the filename generated by the ransomware was used. For a file under

test, if the calculated entropy value of the entire filename string is under six bits then the test passed and the file was considered benign, otherwise, the test failed and the file was considered malicious.

Known File Name Extension Test. As mentioned above, when ransomware encrypts a file it often also tends to change or append an extra extension to the affected file. Sometimes the text of this new extension relates to the name of the ransomware but often the extension is a random string of between three and 50 characters in length. In normal operation, it is very rare that a file's extension is not a well-known value, as typically well-known applications generate files with well-known extensions. This test is aimed at checking and confirming that the extension of the file being written is one of the common extensions [36] [49] [50] [52]. This test uses the collated list, created by the authors, of known extensions which are also used in the *Magic Number Test* described in Section 3.1. If the file extension is present in the list, then it is considered to be well-known. If the file name contains multiple extensions, then this test is applied to the last extension.

The test was then applied to all the files within the test dataset. For a file under test, if the file's extension is well-known then the test passed and the file was considered benign, otherwise, the test failed and the file was considered malicious.

File Name Extension Entropy Test. This test calculates the Shannon [58] entropy value of the file name's extension. If the file has multiple extensions, then the entropy of the entire extension chain is calculated. Often crypto-ransomware will append an extra extension to a file that it has encrypted. This extension can be a text string relating to the ransomware strain, but more recently it has been a random string of between three and 50 characters. When analysing the entropy value of all the extensions in the list of well-known extensions it was found that they all had a Shannon entropy value of below six bits.

The test was then applied to all the files within the test dataset. For a file under test, if the calculated entropy value of the file's extension, or extensions, is below six bytes then the test passed and the file was considered benign, otherwise, the test failed and the file was considered malicious.

3.3. Executable Analysis

The collection of tests described in this section relates to tests performed on the executable code files used to launch the process as well as tests performed on a process's memory captured during its execution. Benign programs were selected that would normally generate files of a specific type. Specific details of the benign programs analysed are provided in **Table B1**. For example, files of type DOCX would usually be created using the Microsoft Word application, so the executable for this application was analysed as well as its memory during its execution.

Strings in Executable Test. Often ransomware executables contain an-

ti-analysis techniques in an attempt to prevent researchers from inspecting the content of the code. These techniques can include obfuscation, polymorphism and encryption of the content of the executable. A consequence of this is that the number of humanly readable strings found within such a file could be significantly lower than would normally be expected. This static analysis technique was applied to both benign as well as ransomware executable files and took the form of extracting strings from the executable and then counting the number and frequency of Windows Application Programming Interface (API) strings that could be identified. This technique has also been leveraged in other ransomware detection systems such as R-PackDroid [66].

No specific metrics, such as the expected number of API strings per KB, are currently available in the literature. So these tests are more exploratory to discover if the type and frequency of API calls differ significantly between ransomware and benign executables and if this measurement would be a useful contributor to a malice score calculation in a ransomware detection system.

Creation and Modification Dates Test. Executable files normally have a significant time interval between when they were placed on the file system and the current execution time. A small interval between the creation date and time and the current date and time could also be used as an indicator of a recently placed malicious program.

This static analysis test was applied to all the executable files shown in the appendix in Table A1 and Table B1. For an executable file under test, if the file's creation or modification date is greater than one day then the test passed and the executable file was considered benign, otherwise, the test failed and the file was considered malicious.

Process Analysis

The following tests could be performed on running processes to determine if any indicators could be identified, that would suggest that the process was malicious. The memory contents of the process under investigation are analysed for indicators of malicious behaviour.

File-less Execution Test. Running processes that do not have an underlying executable on the file system could be considered suspicious as some forms of ransomware execute by being directly injected into memory. These injected programs would then have no underlying executable file present on the file system. This is unusual behaviour for a process and can be used to flag irregular behaviour [67].

This test was applied to the running process. If the process is associated with a file on the file system then the test passed and the process file was considered benign, otherwise, the test failed and the process was considered malicious.

Cryptographic Key Identification Test. The memory and underlying executable file used to launch the process under investigation will be examined for traces of cryptographic keys, as these could indicate that the process is, or will shortly begin, encrypting files. The memory will be searched for keys for the fol-

lowing three cryptographic algorithms: AES [68] [69] [70] [71] [72], Salsa20 [73] and RSA [74] [75]. The AES key testing included checking for the presence of keys of length 128, 192 and 256 bits.

Initially, the executable file that will be used to launch the process with be examined. Subsequently, the memory of the running process will be checked on two occasions, firstly, directly after the process has launched and then subsequently checked again 30 seconds after launch. If no keys are found in each of these tests, then the test passed and the process was considered benign, otherwise, if keys are discovered, the test failed and the process was considered malicious.

Ransom Note Identification Test. The memory of the process under investigation will be examined for traces of typical strings that often appear within ransom notes. These are files normally generated by ransomware programs and are used to inform the user that they have been the victim of a ransomware attack. These files usually contain information on how the user may recover their data. The presence of many keywords close together within the process's memory would be an indicator that the process could be malicious. This test is similar to the previous Ransom Note Creation Test, using the same keywords, however, in this case, it will be performed on the process's memory and not on its output.

This test was applied to the running process. If the process's memory does not contain several examples of the keywords, then the test passed and the process was considered benign, otherwise, the test failed and the process was considered malicious.

Windows API Analysis Test. The memory of the process under investigation will be examined and a review of the number and frequency of all the found window's application programming interface (API) calls will be performed. Executables use these API calls to interact with the operating system and the number and type of calls used together with their frequency will be investigated to determine if this could be used as an indicator that the process under investigation is malicious. This test is similar to the previous Strings in Executable Test, however, in this case, it will be performed on the process's memory and not on the executable file used to launch the process.

No specific metrics, such as the expected number of API strings per KB, are currently available in the literature. So these tests are more exploratory to discover if the type and frequency of API calls differ significantly between ransomware and benign executables and if this measurement would be a useful contributor to a malice score calculation in a ransomware detection system.

3.4. Behaviour Analysis

The actions and behaviour exhibited by the ransomware can also be monitored to identify suspicious behaviour. These tests are outlined below.

Modification of System Restore Points. System restore points are used to recover a system's state or file system files. There are very few occasions where a

process needs to issue commands relating to system restore points, especially concerning their deletion. The state of the system's restore points will be monitored, during the execution of the process under investigation, to determine if they are modified.

This test was applied to the running process. If the systems restore points remained intact two minutes after the launch of the process, then the test passed and the process was considered benign, otherwise, if the restore points had been altered or deleted, the test failed and the process was considered malicious.

Process escalation Some ransomware processes attempt to gain elevated access to resources that are normally protected from an application or user. This is attempted so that the process can gain deeper and broader control of the system and allow them to perform more destructive actions. Identification of such behaviour would prove to be a useful indicator of malicious activity.

This test was applied to the running process. If the running process achieves elevated access or spawns a child process with elevated access then the test fails and the process is considered malicious, otherwise, if the access remains unchanged then the test passed and the process was considered benign.

4. Evaluation and Discussion

The majority of the recorded results for the tests described in Section 3 are provided in Figure 2. The cell colours represent the success of the test and are graded from green to red. 100% pass rate results are represented as a dark green colour, the colour changes depending on the success rate to red which indicates 0% pass rate, or alternatively 100% failure rate. Where the colour does not clearly show the result, then the percentage number is also displayed. Grey indicates that the specific test was not executed on that particular file type. For example, as mentioned above, if the file type should contain a magic number, then this test was performed and the printable character test was ignored.

Some of the tests were exploratory in nature in an attempt to discover if the gathered metrics could be used to identify malicious code. Examples of these exploratory tests were in the cataloguing of API calls distinguishable in the executable as well as the process memory directly after launch and then again 30 seconds after launch. The results of these tests are presented in Figure 3, Figure 4. The remainder of this section reviews the results gathered during the testing and provides some context, discussion and background into the tests and the recorded results. A clarification of a test's success is provided in Table 3.

File content analysis. These tests were performed on the files generated by a process. These tests included the analysis of the created file's magic number value, or for plain text files, the percentage of humanly readable characters within the file was analysed. Other tests included the Chi-Square entropy of the content of the file as well as the BitByte value test. Of all the tests performed these were some of the most successful in differentiating between output generated from benign and malicious processes, a summary of the results is provided in Table 4.

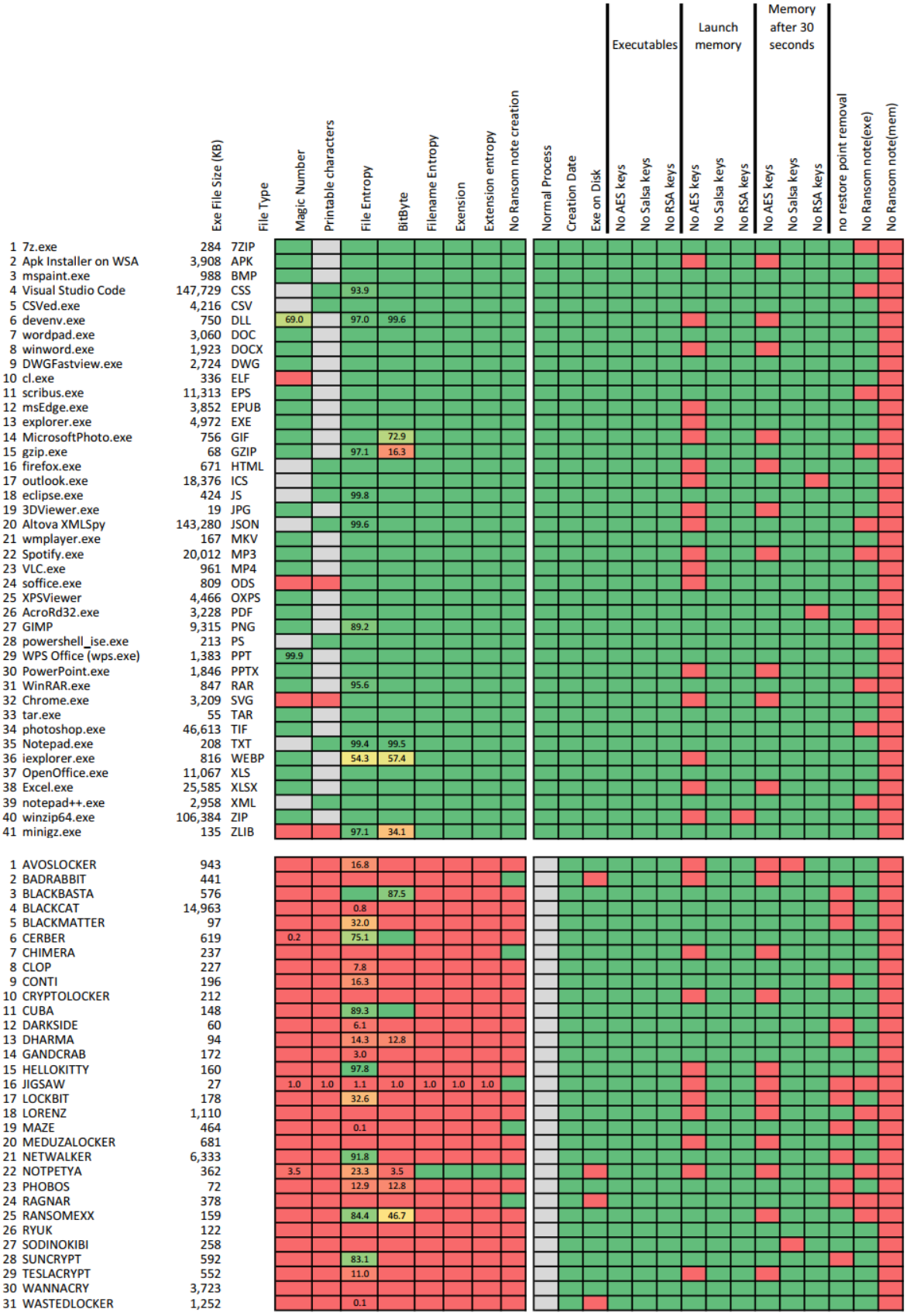

Figure 2. Results overview.

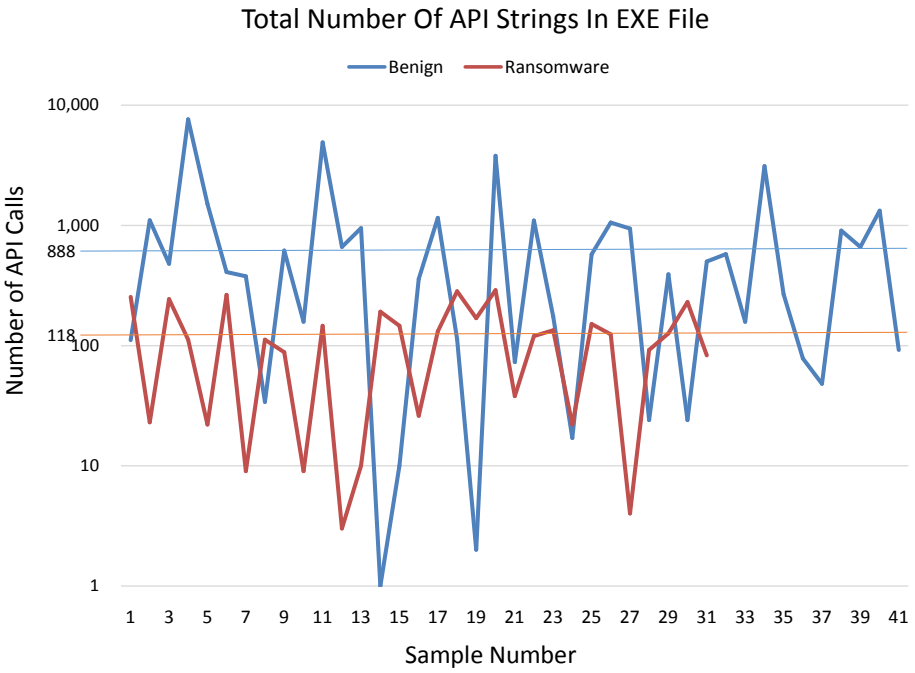

Figure 3. Total API calls in executable.

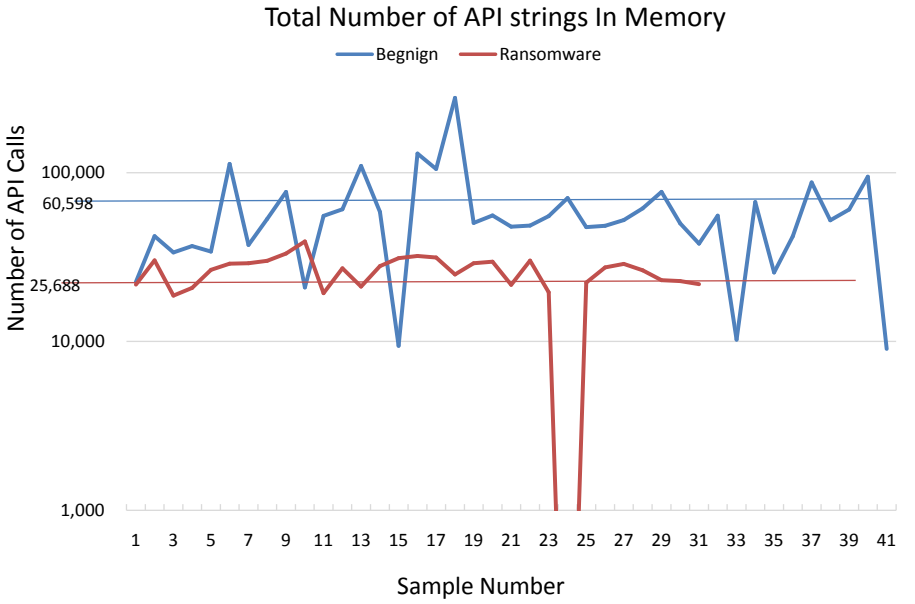

Figure 4. Total API calls in memory.

Table 3. Possible classification outcomes.

Classification	Description
True Positive (TP)	Test passes on benign file
True Negative (TN)	Test fails on ransomware file
False Positive (FP)	Test passes on ransomware file
False Negative (FN)	Test fails on benign file

Table 4. File test performance metrics.

	Accuracy	Recall	Precision	F1
Magic number	0.961	0.998	0.923	0.959
Printable Char.	0.999	0.999	0.999	0.999
File Entropy	0.865	0.831	0.958	0.890
BitByte	0.919	0.914	0.946	0.930
Filename Ent.	0.999	0.999	0.999	0.999
Extension	0.999	0.999	0.999	0.999
Extension Ent.	0.999	0.999	0.999	0.999

No individual test achieved 100% accuracy but the BitByte test is worth highlighting as its results were more accurate than the plain entropy tests when working on files with unknown content. The magic number test combined with known file extension tests also achieved high accuracy, but these rely on the created files having a known extension. These tests could be bypassed, by ransomware using well-known extensions on their output, as highlighted by the results recorded when analysing the files generated by the NotPetya ransomware strain which does not modify the extension of the files it attacks [76].

Generated file name analysis. These tests were performed on the names of the files generated by the process. These tests included the analysis of the entropy of the entire filename, the entropy of the file name's extension as well as validating if the file name's extension was a known value. These tests also achieved high accuracy, a summary of the results is provided in **Table 4**. Contributing factors to this high accuracy was that the benign files used all had well-known file extensions and almost all the tested ransomware strains modified the files name and/or extension in a way that increased the overall entropy of the filename. While the testing did cover more than 30 different ransomware strains, it may not be sufficiently broad enough to generalise this phenomenon. As with some of the file content tests, the exception being the files generated by the NotPetya ransomware strain, which was able to successfully evade this group of tests. This leads us to think that these tests should be applied to a larger test dataset, before generalising the findings.

Ransomnote tests. These tests can also be divided into static and dynamic analysis tests. The static portion of the tests involved examining the executable file used to launch the process and trying to identify several occurrences of typical strings used in ransom notes. This analysis could be performed prior to the launching of the process. In one of the dynamic analysis tests, the running process's volatile memory was examined for the existence of these same ransom note strings. In the other dynamic analysis test, the files generated by the process were examined to determine if the file being created could possibly be a ransom note. No ransom note strings were found in either the begin or ransomware executable binaries. The success rate when looking for ransom note strings within

the memory was very low with positive matches only 20% of the time. These matches were relatively evenly distributed between benign and malicious programs. A consequence of this is that it seems that these metrics would not be suitable for use within a ransomware detection program. The accuracy may be improved for these tests by possibly applying some additional logic to the search, for example, by increasing the dictionary of keywords being searched for, applying natural language processing on the found strings, or analysing the distance between where these words appear and applying a ranking or weighting to the found strings.

The results regarding the dynamic test of analysing the contents of files being created by the process were much more encouraging. No files generated by the benign programs were marked as ransomware, and 80% of the ransom notes generated by the ransomware were successfully identified. Some reasons why this rate was not even higher were that some ransomware strains do not create ransom notes, some ransom notes were actual graphics and some ransomware strains changed the desktop background to display the ransom message. This is a promising finding as many ransomware strains create the ransom note prior to the encryption [63] of the data and a successful interception at this point in the attack would be beneficial.

Identification of cryptographic artefacts. These tests involved attempting to identify cryptographic algorithm artefacts using both static and dynamic analysis methods. The static portion of the tests involved looking for these artefacts in the executable binary files used to launch the process. This analysis could be performed prior to the launching of the process. The dynamic aspect of these tests involved looking for these artefacts in the process's volatile memory, precisely after the process has been launched and then again 30 seconds after the process's launch. The artefacts that were being searched for were AES encryption algorithm keys of length 128 bits, 192 bits and 256 bits, as well as RSA asymmetric and Salsa20 encryption algorithm keys. This resulted in 15 distinct tests per file type resulting in a total of more than 1000 tests being conducted for this group of tests. When reviewing the results it can be seen that no cryptographic artefacts were identifiable in any of the test binaries. Regarding the AES key discovery, then no real pattern could be identified. For benign programs, these keys were identified in 44% of the samples and with ransomware programs, these keys were identified in 35% of the samples. These findings indicate that this metric in its current format is not particularly suited for indicating ransomware activity. It is a known behaviour, that ransomware does use cryptography during its execution, so some explanation for the lack of successful key identification could be that these algorithms are not used in these ransomware samples or that the encryption has either not commenced or has completed when the analysis was performed. The presence of these artefacts in memory of benign programs is not ideal as it complicates the metric.

Behavioural analysis. These tests are aimed at analysing the behaviour of the process under investigation. The idea behind the first test, normal process, was

to try and identify if a running process attempts to alter its execution privileges and try and run as an elevated user. None of the benign processes did this, however, many of the ransomware samples used, would not execute correctly without them being started as the administrator user, which negated the usefulness of this test. No identifying trend could be used to differentiate benign and malicious binary files using the file creation date. With regards to file-less execution, four ransomware samples did spawn a malicious process that had no underlying binary file on disk and approximately 45% of ransomware programs removed system restore points soon after they started executing. Both these last two behaviours were only observed with ransomware programs and could be used as a contributing factor when trying to determine if the process is malicious or not.

Analysis Tests. Some exploratory investigation was also performed in cataloguing and analysing the number and frequency of standard Windows API calls within both the binary executable file (static analysis) as well as the process's volatile memory (dynamic analysis) directly after launch and then again 30 seconds after launch. Note that the Y-axis on the following figures has a logarithmic scale. From Figure 3, it can be seen that the number of API calls present in the executables of benign programs differ by a factor of eight when compared to the number of API calls identified within ransomware programs One possible explanation for this could be that ransomware programs often try and obfuscate their structure prior to execution in an attempt to hinder analysis and a consequence of this being that the API calls are hidden.

To normalise these results, the values were then plotted as a ratio of the number of API calls present divided by the analysed executable file size. These normalised results are shown in Figure 4. The programs were then launched and the volatile memory used by each of these programs was then captured and analysed for Windows API calls. A comparison of the API calls present within each process's memory is presented in Figure 5. Again to aid comparison, the graphs have been normalised by dividing the total number of calls by the size of the total memory being used. Finally, the launched program's memory was captured again 30 seconds after launch and analysed for Windows API calls. A comparison of the API calls present within each process's memory is presented in Figure 6 with the results being normalised.

When reviewing the captured results, it can be seen that the identified API calls within the binary files show signs of possibly being a useful indicator of Windows API obfuscation and thus an indicator of a possibly malicious program. The measurements show that there is an obvious difference between the number of API calls found within the benign and malicious executables. The difference between these two types of executables is not so prominent when analysing the process's volatile memory. However, it is felt by the researchers that these findings merit further investigation. As they stand, using the current metrics, these results would not prove useful as a contributor to the suite of tests used in the malice score calculation. Some refinement of the measurement, such

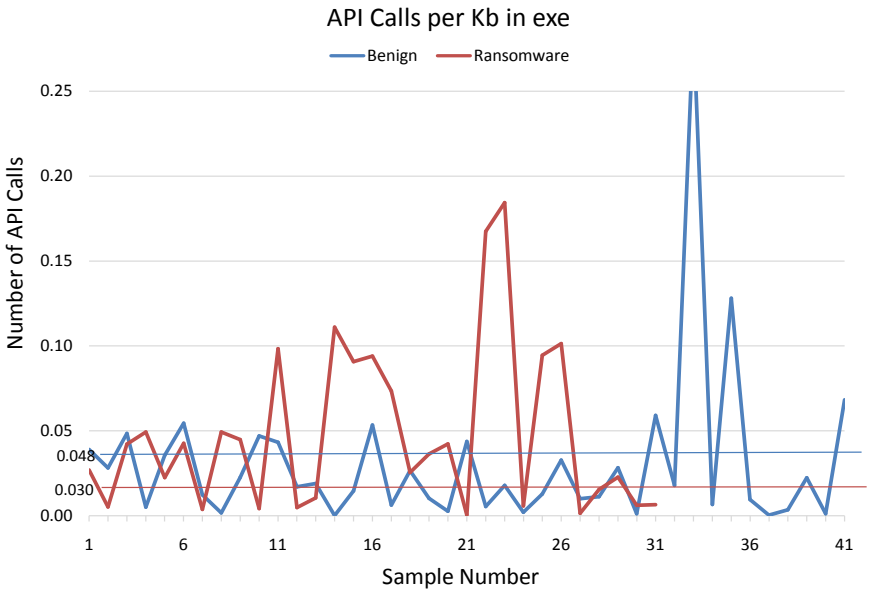

Figure 5. Average API calls in memory per KB at launch.

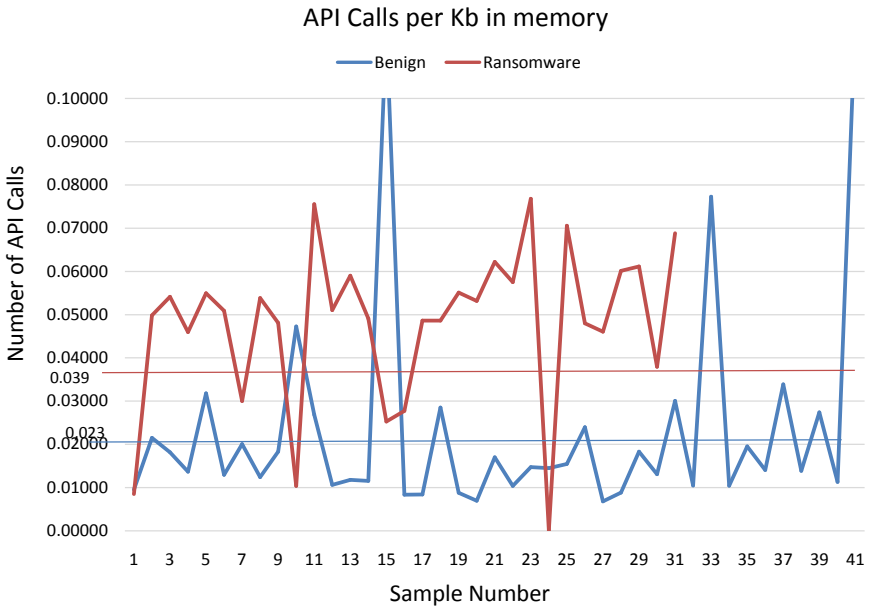

Figure 6. Average API Calls in memory per KB at 30 seconds.

as targeting specific API calls or call frequency analysis may enhance the accuracy of this type of measurement and further investigation into this would be beneficial.

Majority Voting

When reviewing the results from the separate tests mentioned above, it can be seen that several tests achieved a high degree of accuracy in differentiating between benign and malicious programs, using both static and dynamic tests. The results from some tests such as attempting to identify cryptographic artefacts,

ransom note identification within the process and executable or Windows API enumeration, delivered inconclusive results and these tests would require some more investigation, analysis and modification.

It is proposed that a system could be developed that uses a combination of the tests that have been found to be accurate in identifying ransomware. Each test's vote would contribute to an overall malice score for the target file or process, and based on the maximum number of votes the system would classify the target as either malicious or benign. For example, a system could be developed that used the following tests: created file name and extension entropy, well-known extensions, file magic number and printable characters, file content BitByte and entropy values, ransom note creation detection and system restore point removal detection. Based on the findings from this research, a system configured with these tests would have an accuracy of 0.9989 on the dataset used. Some of the interesting test results are highlighted in Figure 7.

When reviewing the results for benign programs shown in Figure 7, it can be seen that the majority of tests consider the processes/files to be benign. Even in cases where some of the individual tests do occasionally give false positives, in all

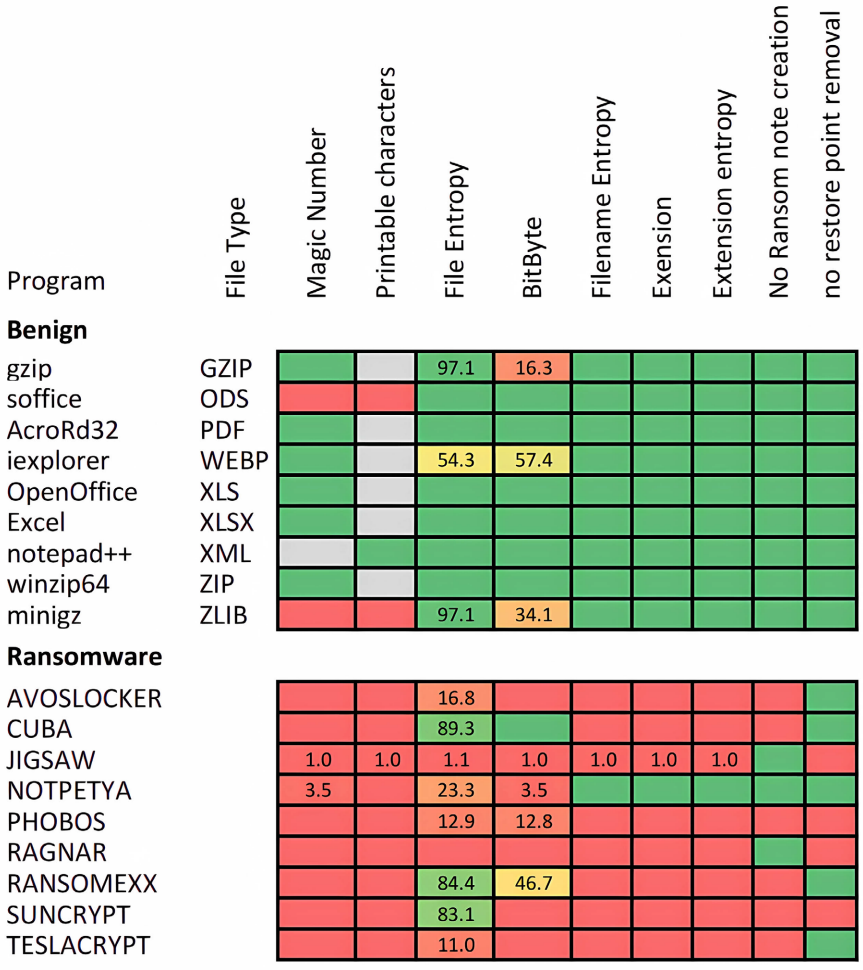

Figure 7. Interesting results.

cases, the majority of the tests vote correctly resulting in a correct overall classification. For example, when looking at the classification for WEBP file types, it can be seen that the individual file entropy and BitByte tests, result in a classification accuracy of around 55%, and 45% of the samples are incorrectly classified as malicious. However, as the remaining six tests correctly vote that the file is benign, these files are ultimately classified as benign. Likewise, when reviewing the result for the ransomware files, in most cases the majority of tests classify the file/process as malicious. The only exception would be on the very rare occasion, the files generated by the Jigsaw ransomware strain, may theoretically receive a false positive classification if the majority of the tests vote that the file/process is benign.

A major strength of the majority voting approach to ransomware detection is that not every test needs to correctly classify a malicious program every time. With equal weighting on the result of each test, it would be sufficient for just a majority of tests to correctly classify the target, for the system to work successfully. Some work could also be performed to investigate whether a weighting or bias could also be applied to the test results meaning that some tests would then have a greater influence on the overall outcome of the classification, than others.

As the detection technique relies on well-known discrete tests, it is also easier for the detection model to be modified, updated and tuned as opposed to a machine learning model where the weightings and strengths of the learned model can be unknown or difficult to influence.

5. Conclusions

This paper proposes a ransomware detection system using a majority voting-based approach. A final *malice score* is derived from the combination of the results from many discrete tests that are conducted on the target process, its executable file or the output that the process generates. These distinct results are then aggregated and used as input for the malice score generation. Based on this score the target is classified as benign or malicious. The paper proposes 23 main tests that could potentially be used in a ransomware detection system with their outcomes, contributing to the overall malice score. The paper also investigates additional potential metrics that could be used in ransomware detection, for example, the presence of Windows API calls in the binary and executing processes' volatile memory.

This research demonstrates that many of the proposed tests achieved a high degree of accuracy in differentiating between benign and malicious targets. The accuracy was then enhanced when a selection of these tests was then combined into a majority voting model. One proposed majority voting model achieves an accuracy of 0.9989. The collaborative approach in generating the final result has many advantages, for example, some individual tests on some occasions may produce incorrect classifications, but the overall accuracy of the detection system as a whole will be unaffected if the majority of the tests produce the correct re-

sults.

As this majority voting detection technique relies on well-known and easily understandable discrete tests, then it is easier for the model to be modified, updated and tuned as opposed to a machine learning approach where the weightings and strengths of the learned model can be unknown or difficult to influence. An additional advantage is that while machine learning models require training, the majority voting approach, proposed in this paper, does not.

5.1. Limitations

While the majority voting approach to identifying malicious processes has a high level of accuracy, as always the situation exists where once a ransomware developer is aware of the techniques being used to identify malicious behaviour, they have the possibility of modifying or adapting the ransomware's behaviour to avoid the tests in newer releases of their programs. The advantage of the majority voting approach is that the system does not rely on a single catch-all test, rather detection is a combination of many accurate tests. A consequence of this is that the ransomware developer may have to significantly modify the behaviour of their programs, and possibly disregard some aspects of their original behaviour to avoid detection.

5.2. Future Work

The results achieved during the Windows API call analysis could possibly be improved by further investigation and modifications to the types of API calls present, their frequency and their position within the file or process memory. One area of further work would be a deeper analysis of this aspect of the binaries and volatile memory. Another area of work would be to introduce a weighting element to the measurements, allowing some tests to have a greater influence on the final classification results.

Analyses of other types of tests could also be performed. Examples of which could be: multiple-file read and write operations, high entropy differences between read and write operations, file tree traversal, privilege escalation, accessing crypto API functionality, accessing unusual domain names, generation of large amounts of traffic, DGA detection [13] [77] and the termination of a large number of processes.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] MalwareBytes (2023) ION Starts Bringing Customers Back Online after LockBit Ransomware Attack.
https://www.malwarebytes.com/blog/news/2023/02/ion-starts-bringing-customers-

back-online-after-lockbit-ransomware-attack

[2] The Telegraph Media Group (2023) Royal Mail Turned down £66m Ransom Demand from Lockbit Hackers. https://www.telegraph.co.uk/business/2023/02/14/royal-mail-turned-66m-ransom-demand-lockbit-hackers/

[3] Oz, H., Aris, A., Levi, A. and Uluagac, A.S. (2021) A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. ArXiv: 2102.06249. http://arxiv.org/abs/2102.06249

[4] Yamany, B., Elsayed, M.S., Jurcut, A.D., Abdelbaki, N. and Azer, M.A. (2022) A New Scheme for Ransomware Classification and Clustering Using Static Features. *Electronics*, **11**, Article No. 3307. https://doi.org/10.3390/electronics11203307

[5] Dutta, N., Jadav, N., Tanwar, S., Sarma, H.K.D. and Pricop, E. (2022) Introduction to Malware Analysis. In: *Cyber Security*: *Issues and Current Trends*. *Studies in Computational Intelligence*, Vol. 995, Springer, Singapore, 129-141. https://doi.org/10.1007/978-981-16-6597-4_7

[6] Lebbie, M., Prabhu, S.R. and Agrawal, A.K. (2022) Comparative Analysis of Dynamic Malware Analysis Tools. In: Dua, M., Jain, A.K., Yadav, A., Kumar, N. and Siarry, P., Eds., *Proceedings of the International Conference on Paradigms of Communication*, *Computing and Data Sciences*, Springer, Singapore, 359-368. https://doi.org/10.1007/978-981-16-5747-4_31

[7] De Gaspari, F., Hitaj, D., Pagnotta, G., De Carli, L. and Mancini, L.V. (2020) THE NAKED SUN: Malicious Cooperation between Benign-Looking Processes. In: Conti, M., Zhou, J., Casalicchio, E. and Spognardi, A., Eds., *Applied Cryptography and Network Security*. *ACNS* 2020. *Lecture Notes in Computer Science*, Vol. 12147, Springer, Cham, 254-274. https://doi.org/10.1007/978-3-030-57878-7_13

[8] Moser, A., Kruegel, C. and Kirda, E. (2007) Limits of Static Analysis for Malware Detection. 23*rd Annual Computer Security Applications Conference* (*ACSAC* 2007), Miami Beach, 10-14 December 2007, 421-430. https://doi.org/10.1109/ACSAC.2007.21

[9] O'Kane, P., Sezer, S. and McLaughlin, K. (2011) Obfuscation: The Hidden Malware. *IEEE Security & Privacy*, **9**, 41-47. https://doi.org/10.1109/MSP.2011.98

[10] Ahmed, M.E., Kim, H., Camtepe, S., Nepal, S. (2021) Peeler: Profiling Kernel-Level Events to Detect Ransomware. In: Bertino, E., Shulman, H. and Waidner, M., Eds., *Computer Security—ESORICS* 2021. *ESORICS* 2021. *Lecture Notes in Computer Science*, Vol. 12972, Springer, Cham, 240-260. https://doi.org/10.1007/978-3-030-88418-5_12

[11] Ahmed, Y.A., Koçer, B. and Al-Rimy, B.A.S. (2020) Automated Analysis Approach for the Detection of High Survivable Ransomware. *KSII Transactions on Internet and Information Systems*, **14**, 2236-2257. https://doi.org/10.3837/tiis.2020.05.021

[12] Kim, G.Y., Paik, J.-Y., Kim, Y. and Cho, E.S. (2022) Byte Frequency Based Indicators for Crypto-Ransomware Detection from Empirical Analysis. *Journal of Computer Science and Technology*, **37**, 423-442. https://doi.org/10.1007/s11390-021-0263-x

[13] Salehi, S., Shahriari, H., Ahmadian, M.M. and Tazik, L. (2018) A Novel Approach for Detecting DGA-Based Ransomwares. 2018 15*th International ISC* (*Iranian Society of Cryptology*) *Conference on Information Security and Cryptology* (*ISCISC*), Tehran, 28-29 August 2018, 1-7. https://doi.org/10.1109/ISCISC.2018.8546941

[14] Scaife, N., Carter, H., Traynor, P. and Butler, K.R. (2016) CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. 2016 *IEEE* 36*th International Confe-*

rence on Distributed Computing Systems (*ICDCS*), Nara, 27-30 June 2016, 303-312. https://doi.org/10.1109/ICDCS.2016.46

[15] Alam, M., Sinha, S., Bhattacharya, S., Dutta, S., Mukhopadhyay, D. and Chattopadhyay, A. (2020) RAPPER: Ransomware Prevention via Performance Counters. ArXiv: 2004.01712. http://arxiv.org/abs/2004.01712

[16] Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S. and Khayami, R. (2020) Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence. *IEEE Transactions on Emerging Topics in Computting*, **8**, 341-351. https://doi.org/10.1109/TETC.2017.2756908

[17] Lokuketagoda, B., Weerakoon, M.P., Kuruppu, U.M., Senarathne, A.N., Yapa Abeywardena, K. (2018) R-Killer: An Email Based Ransomware Protection Tool. 2018 13*th International Conference on Computer Science and Education*, Colombo, 8-11 August 2018, 1-7. https://doi.org/10.1109/ICCSE.2018.8468807

[18] McDonald, G., Papadopoulos, P., Pitropakis, N., Ahmad, J. and Buchanan, W.J. (2022) Ransomware: Analysing the Impact on Windows Active Directory Domain Services. *Sensors*, **22**, Article No. 953. https://doi.org/10.3390/s22030953

[19] Hall, G.A., Hall, G.A. and Davis, W. (2007) Sliding Window Measurement for File Type Identification. https://api.semanticscholar.org/CorpusID:14149550

[20] Lee, K., Lee, S.-Y. and Yim, K. (2019) Effective Ransomware Detection Using Entropy Estimation of Files for Cloud Services. In: Esposito, C., Hong, J. and Choo, K.-K., Eds., *Pervasive Systems, Algorithms and Networks. I-SPAN* 2019. *Communications in Computer and Information Science*, Vol. 1080, Springer, Cham, 133-139. https://doi.org/10.1007/978-3-030-30143-9_11

[21] Lee, K., Lee, S.-Y. and Yim, K. (2019) Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems. *IEEE Access*, **7**, 110205-110215. https://doi.org/10.1109/ACCESS.2019.2931136

[22] VandenBrink, R. (2016) Using File Entropy to Identify "Ransomwared" Files. https://isc.sans.edu/forums/diary/Using+FileEn-tropy+to+Identify+Ransomwared+Files/21351/

[23] Al-Rimy, B.A.S., Maarof, M.A. and Shaid, S.Z.M. (2019) Crypto-Ransomware Early Detection Model Using Novel Incremental Bagging with Enhanced Semi-Random Subspace Selection. *Future Generation Computer Systems*, **101**, 476-491. https://doi.org/10.1016/j.future.2019.06.005

[24] Bottazzi, G., Italiano, G.F. and Spera, D. (2018) Preventing Ransomware Attacks through File System Filter Drivers. *Proceedings of the* 2*nd Italian Conference on Cyber Security* (*ITASEC*18), Milan, 6-9 February 2018. https://www.researchgate.net/publication/323125541_Preventing_Ransomware_Attacks_Through_File_System_Filter_Drivers

[25] Ki, Y., Kim, E. and Kim, H.K. (2015) A Novel Approach to Detect Malware Based on API Call Sequence Analysis. *International Journal of Distributed Sensor Networks*, **11**. https://doi.org/10.1155/2015/659101

[26] Song, S., Kim, B. and Lee, S. (2016) The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform. *Mobile Information Systems*, **2016**, Article ID: 2946735. https://doi.org/10.1155/2016/2946735

[27] Berrueta, E., Morato, D., Magana, E. and Izal, M. (2019) A Survey on Detection Techniques for Cryptographic Ransomware. *IEEE Access*, **7**, 144925-144944. https://doi.org/10.1109/ACCESS.2019.2945839

[28] Dargahi, T., Dehghantanha, A., Bahrami, P.N., Conti, M., Bianchi, G. and Benedetto, L. (2019) A Cyber-Kill-Chain Based Taxonomy of Crypto-Ransomware Features.

Journal of Computer Virology and Hacking Techniques, **15**, 277-305. https://doi.org/10.1007/s11416-019-00338-7

[29] Maigida, A.M., Abdulhamid, S.M., Olalere, M., Alhassan, J.K., Chiroma, H. and Dada, E.G. (2019) Systematic Literature Review and Metadata Analysis of Ransomware Attacks and Detection Mechanisms. *Journal of Reliable Intelligent Environments*, **5**, 67-89. https://doi.org/10.1007/s40860-019-00080-3

[30] Kharraz, A. and Kirda, E. (2017) Redemption: Real-Time Protection against Ransomware at End-Hosts. In: Dacier, M., Bailey, M., Polychronakis, M. and Antonakakis, M., Eds., *Research in Attacks, Intrusions, and Defenses. RAID* 2017. *Lecture Notes in Computer Science*, Vol. 10453, Springer, Cham, 98-119. https://doi.org/10.1007/978-3-319-66332-6_5

[31] Abbasi, M.S., Al-Sahaf, H. and Welch, I. (2021) Automated Behavior-Based Malice Scoring of Ransomware Using Genetic Programming. 2021 *IEEE Symposium Series on Computational Intelligence*, Orlando, 5-7 December 2021. https://doi.org/10.1109/SSCI50451.2021.9660009

[32] Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S. and Maggi, F. (2016) ShieldFS: A Self-Healing, Ransomware-Aware Filesystem. *Proceedings of the 32nd Annual Conference on Computer Security Applications*, Los Angeles, 5-8 December2016, 336-347. https://doi.org/10.1145/2991079.2991110

[33] John, T.C., Abbasi, M.S., Al-Sahaf, H. and Welch, I. (2022) Automatically Evolving Malice Scoring Models through Utilisation of Genetic Programming: A Cooperative Coevolution Approach. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Boston, 9-13 July 2022, 562-565. https://doi.org/10.1145/3520304.3529063

[34] Kharaz, A., Arshad, S., Mulliner, C., Robertson, W. and Mulliner, C. (2016) UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. 25*th USENIX Security Symposium* (*USENIX Security* 16), Austin, 10-12 August 2016, 757-772. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz

[35] Mehnaz, S., Mudgerikar, A. and Bertino, E. (2018) RWGuard: A Real-Time Detection System against Cryptographic Ransomware. In: Bailey, M., Holz, T., Stamatogiannakis, M. and Ioannidis, S., Eds., *Research in Attacks, Intrusions, and Defenses. RAID* 2018. *Lecture Notes in Computer Science*, Vol. 11050, Springer, Cham, 114-136. https://doi.org/10.1007/978-3-030-00470-5_6

[36] Davies, S.R., Macfarlane, R. and Buchanan, W.J. (2022) NapierOne: A Modern Mixed File Data Set Alternative to Govdocs1. *Forensic Science International: Digital Investigation*, **40**, Article ID: 301330. https://doi.org/10.1016/j.fsidi.2021.301330

[37] McIntosh, T., Jang-Jaccard, J., Watters, P. and Susnjak, T. (2019) The Inadequacy of Entropy-Based Ransomware Detection. In: Gedeon, T., Wong, K. and Lee, M., Eds., *Neural Information Processing. ICONIP* 2019. *Communications in Computer and Information Science*, Vol. 1143, Springer, Cham, 181-189. https://doi.org/10.1007/978-3-030-36802-9_20

[38] Ganfure, G.O., Wu, C.-F., Chang, Y.-H. and Shih, W.-K. (2020) DeepGuard: Deep Generative User-Behavior Analytics for Ransomware Detection. 2020 *IEEE International Conference on Intelligence and Security Informatics*, Arlington, 9-10 November 2020, 181-189. https://doi.org/10.1109/ISI49825.2020.9280508

[39] Manavi, F. and Hamzeh, A. (2022) A Novel Approach for Ransomware Detection Based on PE Header Using Graph Embedding. *Journal of Computer Virology and Hacking Techniques*, **18**, 285-296. https://doi.org/10.1007/s11416-021-00414-x

[40] Prachi and Kumar, S. (2022) An Effective Ransomware Detection Approach in a Cloud Environment Using Volatile Memory Features. *Journal of Computer Virology and Hacking Techniques*, **18**, 407-424. https://doi.org/10.1007/s11416-022-00425-2

[41] Sheen, S., Asmitha, K.A. and Venkatesan, S. (2022) R-Sentry: Deception Based Ransomware Detection Using File Access Patterns. *Computers and Electrical Engineering*, **103**, Article ID: 108346. https://doi.org/10.1016/j.compeleceng.2022.108346

[42] De Gaspari, F., Hitaj, D., Pagnotta, G., De Carli, L. and Mancini, L.V. (2022) Evading Behavioral Classifiers: A Comprehensive Analysis on Evading Ransomware Detection Techniques. *Neural Computing and Applications*, **34**, 12077-12096. https://doi.org/10.1007/s00521-022-07096-6

[43] Lee, J. and Lee, K. (2022)A Method for Neutralizing Entropy Measurement-Based Ransomware Detection Technologies Using Encoding Algorithms. *Entropy*, **24**, Article No. 239. https://doi.org/10.3390/e24020239

[44] Scaife, N., Carter, H., Traynor, P. and Butler, K.R. (2016)CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. 2016 *IEEE* 36*th International Conference on Distributed Computing Systems*, Nara, 27-30 June 2016. https://doi.org/10.1109/ICDCS.2016.46

[45] Gharib, A. and Ghorbani, A. (2017) DNA-Droid: A Real-Time Android Ransomware Detection Framework. In: Yan, Z., Molva, R., Mazurczyk, W. and Kantola, R., Eds., *Network and System Security. NSS* 2017. *Lecture Notes in Computer Science*, Vol. 10394, Springer, Cham, 184-198. https://doi.org/10.1007/978-3-319-64701-2_14

[46] Davies, S.R., Macfarlane, R. and Buchanan, W.J. (2021) NapierOne. http://napierone.com/Website/index.html

[47] Nieuwenhuizen, D. (2017) A Behavioural-Based Approach to Ransomware Detection. https://api.semanticscholar.org/CorpusID:20947416

[48] Wikipedia. List of File Formats. https://en.wikipedia.org/wiki/Listoffilefor-mats

[49] Buchanan. Digital Forensics Magic Numbers. https://asecuritysite.com/forensics/magic

[50] Google (2015) File Types Indexable by Google. https://support.google.com/webmasters/answer/35287?hl=en

[51] Kessler, G. GCK'S File Signature Table. https://www.garykessler.net/library/file_sigs.html

[52] Leommoore. File Magic Numbers. GitHub. https://gist.github.com/leommoore/f9e57ba2aa4bf197ebc5

[53] Wikipedia. List of File Signatures. https://en.wikipedia.org/wiki/List_of_file_signatures

[54] Genç, Z.A., Lenzini, G. and Ryan, P.Y.A. (2018) No Random, No Ransom: A Key to Stop Cryptographic Ransomware. In: Giuffrida, C., Bardin, S. and Blanc, G., Eds., *Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA* 2018. *Lecture Notes in Computer Science*, Vol. 10885, Springer, Cham, 234-255. https://doi.org/10.1007/978-3-319-93411-2_11

[55] Genç, Z.A., Lenzini, G. and Ryan, P.Y.A. (2020) NOCRY: No More Secure Encryption Keys for Cryptographic Ransomware. In: Saracino, A. and Mori, P., Eds., *Emerging Technologies for Authorization and Authentication. ETAA* 2019. *Lecture Notes in Computer Science*, Vol. 11967, Springer, Cham, 69-85.

https://doi.org/10.1007/978-3-030-39749-4_5

[56] Kharraz, A. and Kirda, E. (2017) Redemption: Real-Time Protection against Ransomware at End-Hosts. In: Dacier, M., Bailey, M., Polychronakis, M. and Antonakakis, M., Eds., *Research in Attacks, Intrusions, and Defenses. RAID* 2017. *Lecture Notes in Computer Science*, Vol. 10453, Springer, Cham, 98-119. https://doi.org/10.1007/978-3-319-66332-6_5

[57] Choudhury, P., Kumar, K.R.P., Nandi, S. and Athithan, G. (2019) An Empirical Approach towards Characterization of Encrypted and Unencrypted VoIP Traffic. *Multimedia Tools and Applications*, **79**, 603-631. https://doi.org/10.1007/s11042-019-08088-w

[58] Shannon, C.E. (1948) A Mathematical Theory of Communication. *The Bell System Technical Journal*, **27**, 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[59] Karl Pearson, F.R.S. (2009) X. *On the Criterion That a Given System of Deviations From the Probable in the Case of a Correlated System of Variables is Such That It Can Be Reasonably Supposed to Have Arisen From Random Sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **50**, 157-175. https://www.tandfonline.com/doi/abs/10.1080/14786440009463897 https://doi.org/10.1080/14786440009463897

[60] Davies, S.R., Macfarlane, R. and Buchanan, W.J. (2022) Comparison of Entropy Calculation Methods for Ransomware Encrypted File Identification. *Entropy*, **24**, Article No. 1503. https://doi.org/10.3390/e24101503

[61] Walker, J. (2008) A Pseudorandom Number Sequence Test Program. Pseudorandom Number Sequence Test Program. https://www.fourmilab.ch/random/

[62] Davies, S.R., Macfarlane, R. and Buchanan, W.J. (2021) Differential Area Analysis for Ransomware Attack Detection within Mixed File Datasets. *Computers and Security*, **108**, Article ID: 102377. https://doi.org/10.1016/j.cose.2021.102377

[63] Lemmou, Y., Lanet, J.-L. and Souidi, E.M. (2021) In-Depth Analysis of Ransom Note Files. *Computers*, **10**, Article No. 145. https://doi.org/10.3390/computers10110145

[64] Andronio, N., Zanero, S. and Maggi, F. (2015) HELDROID: Dissecting and Detecting Mobile Ransomware. In: Bos, H., Monrose, F. and Blanc, G., Eds., *Research in Attacks, Intrusions, and Defenses. RAID* 2015. *Lecture Notes in Computer Science*, Vol. 9404, Springer, Cham, 382-404 https://doi.org/10.1007/978-3-319-26362-5_18

[65] Li, W.-J., Wang, K., Stolfo, S.J. and Herzog, B. (2005) Fileprints: Identifying File Types by N-Gram Analysis. *Proceedings from the 6th Annual IEEE SMC Information Assurance Workshop*, West Point, 15-17 June 2005, 64-71. https://doi.org/10.1109/IAW.2005.1495935

[66] Scalas, M., Maiorca, D., Mercaldo, F., Visaggio, C.A., Martinelli, F. and Giacinto, G. (2018) R-PackDroid: Practical on-Device Detection of Android Ransomware. https://www.researchgate.net/publication/325358530_R-PackDroid_Practical_On-Device_Detection_of_Android_Ransomware

[67] Kara, I. (2023) Fileless Malware Threats: Recent Advances, Analysis Approach through Memory Forensics and Research Challenges. *Expert Systems with Applications*, **214**, Article ID: 119133. https://doi.org/10.1016/j.eswa.2022.119133 https://www.sciencedirect.com/science/article/pii/S0957417422021510

[68] Balogh, Š. and Pondelik, M. (2011) Capturing Encryption Keys for Digital Analysis. *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, Prague, 15-17 September 2011, 759-763. https://doi.org/10.1109/IDAACS.2011.6072872

[69] Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J. and Felten, E.W. (2009) Lest We Remember: Cold-Boot Attacks on Encryption Keys. *Communications of the ACM*, **52**, 91-98. https://doi.org/10.1145/1506409.1506429

[70] Maartmann-Moe, C., Thorkildsen, S.E. and Årnes, A. (2009) The Persistence of Memory: Forensic Identification and Extraction of Cryptographic Keys. *Digital Investigation*, **6**, S132-S140. https://doi.org/10.1016/j.diin.2009.06.002

[71] Heninger, N. and Feldman, A. (2008) AESKeyFind. https://github.com/makomk/aeskeyfind

[72] Kornblum, J. (2017) Findaes. https://sourceforge.net/u/jessekornblum/profile/

[73] de Loaysa Babiano, L.F., Macfarlane, R. and Davies, S.R. (2023) Evaluation of live forensic techniques, towards Salsa20-Based cryptographic ransomware mitigation. *Forensic Science International: Digital Investigation*, **46**, Article ID: 301572. https://doi.org/10.1016/j.fsidi.2023.301572

[74] Joseph, P. and Norman, J. (2020) Systematic Memory Forensic Analysis of Ransomware Using Digital Forensic Tools. *International Journal of Natural Computing Research*, **9**, 61-81. https://doi.org/10.4018/IJNCR.2020040105

[75] Klein, T. (2006) All Your Private Keys Are Belong to Us. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=cf85042cca0da125b860db7c2fefb38012396cbc

[76] Sai, R.L.P. and Kumar, T.P. (2019) Reverse Engineering the Behaviour of NotPetya Ransomware. *International Journal of Recent Technology and Engineering*, **7**, 574-578. https://www.ijrte.org/wp-content/uploads/papers/v7i6s/F03120376S19.pdf

[77] Chadha, S. and Kumar, U. (2017) Ransomware: Let's Fight Back! 2017 *International Conference on Computing, Communication and Automation* (*ICCCA*), Greater Noida, 5-6 May 2017, 925-930. https://doi.org/10.1109/CCAA.2017.8229926

Appendix

Table A1. SHA256 hashes of ransomware strains used.

Strain	SHA256 Hash
AVOSLOCKER	718810b8eeb682fc70df602d952c0c83e028c5a5bfa44c506756980caf2edebb
BADRABBIT	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da
BLACKBASTA	5d2204f3a20e163120f52a2e3595db19890050b2faa96c6cba6b094b0a52b0aa
BLACKCAT	847fb7609f53ed334d5affbb07256c21cb5e6f68b1cc14004f5502d714d2a456
BLACKMATTER	be5bc29f58b868f4ff8cd66b4526535593e515a697bb8951c625bdfed13cccb7
CERBER	e67834d1e8b38ec5864cfa101b140aeaba8f1900a6e269e6a94c90fcbfe56678
CHIMERA	1dacdc296fd6ef6ba817b184cce9901901c47c01d849adfa4222bfabfed61838
CLOP	a867deb1578088d066941c40e598e4523ab5fd6c3327d3afb951073bee59fb02
CONTI	2fc6d7df9252b1e2c4eb3ad7d0d29c188d87548127c44cebc40db9abe8e5aa35
CRYPTOLOCKER	5e902a138174c34e5445685c82b2044e0b35565854471aaccef0315c77288dc9
CUBA	936119bc1811aeef01299a0150141787865a0dbe2667288f018ad24db5a7bc27
DARKSIDE	508dd6f7ed6c143cf5e1ed6a4051dd8ee7b5bf4b7f55e0704d21ba785f2d5add
DHARMA	c2ab289cbd2573572c39cac3f234d77fdf769e48a1715a14feddaca8ae9d9702
GANDCRAB	64d341ecbc52f9d78080bf23559ec1778824979dd19498ee44032ec1d5224ff6
HELLOKITTY	501487b025f25ddf1ca32deb57a2b4db43ccf6635c1edc74b9cff54ce0e5bcfe
JIGSAW	3d8d58e8b7431c871121c2d3669f68054eda11cbfb735a3ab689734d4544dab3
LOCKBIT	43ced481e0f68fe57be3246cc5aede353c9d34f4e15d0afe443b5de9514d3ce4
LORENZ	1264b40feaa824d5ba31cef3c8a4ede230c61ef71c8a7994875deefe32bd8b3d
MAZE	3885589a3c94d0475a6d994e4644e682f4cff93f8b4d65f37508ffe706861363
MEDUSALOCKER	2e9fceb91d4378a4e67250f0cb633a020be6eb1c57237272a50cb4db36997db7
NETWALKER	57cf4470348e3b5da0fa3152be84a81a5e2ce5d794976387be290f528fa419fd
NOTPETYA	b53f3c0cd32d7f20849850768da6431e5f876b7bfa61db0aa0700b02873393fa
PHOBOS	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2
RAGNAR	5469182495d92a5718e0e1dcdf371e92b79724e427050154f318de693d341c89
RANSOMEXX	4cae449450c07b7aa74314173c7b00d409eabfe22b86859f3b3acedd66010458
RYUK	d083ecc1195602c45d9cb75a08c395ad7d2b0bf73d7e70e2fc76101c780dd38f
SODINOKIBI	06b323e0b626dc4f051596a39f52c46b35f88ea6f85a56de0fd76ec73c7f3851
SUNCRYPT	0d7ed584dd1ae3cc071ad1b2400a5c534d19206be7a98a6046959a7267c063a1
TESLACRYPT	4de6675c089aad8a52993b1a21afd06dc7086f4ea948755c09a7a8471e4fddbd
WANNACRY	32f24601153be0885f11d62e0a8a2f0280a2034fc981d8184180c5d3b1b9e8cf
WASTEDLOCKER	905ea119ad8d3e54cd228c458a1b5681abc1f35df782977a23812ec4efa0288a

Table B1. Details of benign programs used.

Program Name	File Type	Version
7z.exe	7ZIP	9.2
Apk Installer on WSA	APK	Apk Installer 4.7
mspaint.exe	BMP	Version 20H2 19042.1466
Visual Studio Code	CSS	Visual Studio Code v 1.73.1
CSVed.exe	CSV	Csved 2.5.6
devenv.exe	DLL	Version 4.8.04084
wordpad.exe	DOC	Version 20H2 19042.1466
winword.exe	DOCX	Version 15.0.4430.1017
DWGFastview.exe(gcad)	DWG	build 221110 - 32bit V6.0.0
cl.exe	ELF	Version 19.29.30139
scribus.exe	EPS	Version 1.4.8
msEdge.exe	EPUB	Version 106.0.1370.52
explorer.exe	EXE	22H2 (10.0.22621.675)
MicrosoftPhoto.exe	GIF	Microsoft Photos 2022.30070.26007.0
gzip.exe	GZIP	Version 1.3.12
firefox.exe	HTML	105.0.3
outlook.exe	ICS	Version 15.0.4430.1017
eclipse.exe	JS	Version: 2020-03 (4.15.0) uild id: 20200313-1211
3DViewer.exe	JPG	7.2107.7012.0
Altova XMLSpy	JSON	Version 2023
wmplayer.exe	MKV	12.0.22621.457
Spotify.exe	MP3	1.1.95.893.g6cf4d40c
VLC.exe	MP4	Version 3.0.16
soffice.exe	ODS	7.4.3.3
XPSViewer	OXPS	Version 20H2 19042.1466
AcroRd32.exe	PDF	Version 22.768
GIMP	PNG	Version 2.10
powershell_ise.exe	PS	5.1.19041.1320
WPS Office (wps.exe)	PPT	11.2.0.11388
PowerPoint.exe	PPTX	Version 15.0.4430.1017
WinRAR.exe	RAR	3.42
Chrome.exe	SVG	Version 106.0.5249.119
tar.exe	TAR	bsdtar 3.5.2 - libarchive 3.5.2 zlib/1.2.5.f-ipp
photoshop.exe	TIF	13.1-2x32
Notepad.exe	TXT	Version 20H2 19042.1466
iexplorer.exe	WEBP	11.630.19041.0
OpenOffice.exe	XLS	4.1.3
Excel.exe	XLSX	Version 15.0.4430.1017
notepad++.exe	XML	7.9.1
winzip64.exe	ZIP	27.0 15240
minigz.exe	ZLIB	Version 1