Prognosis Negative: Evaluating Real-Time Behavioral Ransomware Detectors

Abhinav Gupta
University of Colorado Boulder
abhinav.n.gupta@colorado.edu

Aditi Prakash
University of Colorado Boulder
aditi.prakash@colorado.edu

Nolen Scaife
University of Colorado Boulder
scaife@colorado.edu

***Abstract*—Ransomware attacks continue to grow in both severity and prevalence as attackers target public infrastructure, governments, and companies. This problem has spawned a wide range of anti-ransomware techniques which are evaluated by their capability of detecting known samples, but these samples are difficult to obtain and unreliable. Due to polymorphism and the ease of implementing new methods of attack, anti-ransomware is more likely needed to protect against an unknown binary than a known sample. In this paper, we create an extensible framework (Farfel) for testing anti-ransomware products that does not rely on evaluation with known samples. Our framework consists of 21 unique fundamental behavior variations which can be selected to form 1,536 unique attacks. Using this, we evaluate seven commercial and academic anti-ransomware products to identify gaps in coverage of fundamental behaviors. Our evaluation uncovers significant gaps in every product, with three commercial products detecting zero cases. We believe that testing on a wide variety of behaviors will ultimately result in more effective detectors in the future.**

1. Introduction

The past 15 years have seen explosive growth in crypto-ransomware[1] attacks [11], [43], [86]; in these attacks, the victim's data is encrypted until the victim pays the required ransom for recovery. Fueled by advances in pseudo-anonymous payments [74], anonymity networks [67], [68], ransomware-as-a-service [84], and botnets [14], [98], the operators of these systems generally operate with impunity.

Since GPCode in 2005 [84], such ransomware has grown to a multi-million dollar business [79], [90]. The severity of these attacks is increasing and is now not limiting to financial loss; attackers are now targeting public services such as hospitals, law enforcement, and critical infrastructure [15], [94].

To counter the severity of this problem, both academia and industry continue to investigate and deploy anti-ransomware technologies. Commercial products are often marketed as providing protection against unknown ransomware threats. Similarly, academic researchers report high true positive and low false positive rates in their evaluations for detecting known ransomware. However, the increasing prevalence of polymorphic malware [32] implies that anti-ransomware is more likely to need to protect against an unknown binary than a known sample. In this paper, we answer the question *"Why does ransomware detection remain a problem despite the the availability of highly effective (as reported) behavioral ransomware detectors?"* and make the following contributions:

- **Characterization of Fundamental Behaviors:** We begin by characterizing the fundamental (i.e., required) behaviors for ransomware attacks. We then show how some strategies for detection target discretionary (i.e., not required) behaviors, which will inevitably lead to false negatives.
- **Identification of Problematic Evaluations:** We find that the common evaluation methodology for anti-ransomware works relies on finding or creating working samples. The challenge in doing so forces authors to perform a best effort analysis with samples that may overlap in behavior. This leads to insufficient testing and a reliance on seen behaviors rather than possible behaviors.
- **Development of a Reliable Testing Framework:** We develop the first extensible testing framework, Farfel, for anti-ransomware developers that allows the user to select individual behaviors to compose complete ransomware attack. To test the effectiveness of Farfel, we evaluate two academic and five commercial ransomware detectors using 1,536 unique attacks and find that no technology successfully detects all attacks. Worse still, three commercial products detect none of the attacks.

Our results show that it is trivial to evade anti-ransomware products using specific behaviors, leading to continued growth of this problem. Rather than measuring the success of a detector at alerting on known samples, we instead propose an alternate strategy: identify gaps in behavioral coverage using generated test cases composed of a wide range of behaviors. Our intent is to make Farfel and its source available to researchers upon request and accept contributions of new behavior variations. Our hope is that our community will embrace an adaptable and reliable testing framework for future works. We believe that improved evaluation will lead to improved detectors.

The remainder of this paper is structured as follows: In Section 2, we discuss present related work; in Section 3, we provide a characterization of ransomware, its behaviors, and anti-ransomware strategies; Section 4 describes the two major practical challenges in developing a robust anti-ransomware solution; in Section 5 we describe the design of our evaluation framework, Farfel; in Section 6 we evaluate existing detectors using our framework and provide the results; Section 7 offers discussion on ran-

1. We refer to crypto-ransomware as ransomware in the remainder of this paper.

somware detection, commercial simulators, and future work; we conclude in Section 8.

2. Related Work

Previous ransomware research has primarily focused on the problem of creating detectors which trigger when expected behaviors occur [22], [46], [85]. There has been such an effort to produce ransomware detectors that surveying both ransomware and detection techniques is a major research task [6], [12], [24], [67]. Despite the variety of techniques, ransomware continues to be a major problem [11].

To aid in evaluating the efficacy of malware detectors, researchers have continued to rely on both real-world and synthetic samples [30], [36], [71], [85]. The dearth of working, reproducible ransomware samples [6], [12], [22], [49], [85] forces researchers to either scavenge for working samples [13] or develop their own [36]. The result is that each work is evaluated on a set of curated tests which makes it challenging to identify gaps in these detectors or compare their evaluations. We examine how this manifests in existing anti-ransomware works in Section 4.

A commercial ransomware tester, RanSim [52], allows consumers to test their own systems for ransomware susceptibility; we discuss its unsuitability for scientific research in Section 7.4. Closest to our work is that of Christodorescu and Jha [19], who procedurally generated tests for malware detectors in order to evaluate their effectiveness at detecting obfuscated samples. Like their work, we find that the effectiveness of existing detectors is poor, but more importantly, we present a principled and expandable approach to evaluating ransomware detectors for future works.

3. Characterization of Ransomware and Defenses

Ransomware is a type of extortion attack in which a victim's resources are made inaccessible until a payment is made. While this attack can be made successful in various forms, this work is solely concerned with *crypto-ransomware*. In this attack, the victim's documents are encrypted and a payment is required to acquire the decryption key. When the attack is performed correctly (i.e., with strong encryption implemented well) and in the absence of mitigating factors (e.g., backups), the only way to recover is to pay. Young and Yung discuss early such attacks dating back to the 1980s and define desirable (for the attacker) properties of ransomware (called cryptovirological attacks in the paper) [101]. Informally, the attacker desires to maintain control over a set of resources in such a way that if the virus is removed, the access is permanently irrecoverable. In the case of ransomware, this property of survivability is discretionary to the attacker such that if the attacker chooses, the attack is reversible.

3.1. Fundamental Behaviors

The goals of ransomware stem from [101]: survivability and reversibility. In particular, the goal of the attacker is to establish and maintain reversible control over the victim's data; without these properties there is little incentive for the victim to pay. Note that reversibility is not necessary for extortion; the attacker could simply destroy the files, extort a payment, and provide no relief to the victim. However, ransomware authors have shown interest in proving that this is not the case, including building elaborate customer service sites and offering free sample decryption [27], [54], [67], [85].

Ransomware must, at a minimum, perform some behaviors in service of its goal of extorting the user. In [85], [87], the authors noted that the actions which ransomware must perform are file reading, writing, and destruction. While generally correct, this characterization can be extended into several critical behaviors. We call these the *fundamental behaviors* of ransomware:

Data Discovery. To identify candidate data for encryption, the sample must, for example, traverse the filesystem, contain a known list of files to encrypt at execution, or discover raw disks.

Data Selection. A policy for which data will be accessed and which will not must be included. For example, the sample might encrypt PDF documents and any other files over 100KB in the user's home directory.

Input Strategy. As noted in [85], the sample must read the data. However, the sections of the data to be read, the number of bytes in each read, the reading method (e.g., raw disk or filesystem), and the time between such I/O requests can be varied. If the sample will only encrypt the middle 50% of a file, then only that portion of the file must be read.

Encryption. The sample must select a cryptographic algorithm, key, and implementation to be applied to the input. In general, the specific strategy for obtaining the key (e.g., hardcoded or via the Internet) or generating it is not fundamental. Ransomware need not use the platform's cryptographic APIs to generate keys or encrypt data, for example. Samples could arrive with their own functions or statically-linked libraries.

Output Strategy. Once the data has been read and encrypted in-memory, the output strategy determines how the content is written back to the disk (and subsequently the original data is destroyed). The sample could, for example, overwrite the original data in place or write into a separate file and delete the original.

The fundamental behaviors have no bearing on the delivery of the attack. For example, ransomware could be delivered as a document macro, injected into a running process during an exploit, typed at a command interpreter, or downloaded and executed. Regardless of the implementation, the fundamental properties are required. However, attackers have flexibility in how these behaviors are implemented. A sample could choose a file discovery algorithm based on the time of day, as a simple example. Such behavior would make independent executions of the same sample non-deterministic. This is one of the challenges in developing strong anti-ransomware techniques, as we will discuss later in this section and the paper.

3.2. General Behaviors

The fundamental behaviors are a subset of the *general behaviors* of ransomware. General behaviors which are not also fundamental are discretionary. They are not required to meet the goals of survivability and reversibility, and as such, ransomware does not need to exhibit these behaviors to be successful. Our goal is not to provide an exhaustive list of all behaviors seen in the literature; instead, we provide a sample of commonly-referenced behaviors:

Ransom Message. These messages, which form the contact point between the attacker and the victim for extortion, are not required. The attacker may choose to provide an out-of-band (e.g., ransom e-mail [58]) message or no message at all. In large-scale attacks, the attacker may not provide a message on individual hosts, but rather send one message through a different channel.

Execution from File. The attack does not need to be executed from an independent "ransomware" executable. The code could be injected into a benign process during an exploit or entered into an interpreter, for example.

Network Communication. Many ransomware samples attempt to reach out to a server to obtain a public key for encryption. Although this is a relatively common practice, it is not required (i.e., since the key could be hardcoded) and therefore not fundamental.

File Extension Change. Some samples change the file extension on targeted files to indicate success to the victim. Such a behavior is not fundamental because it does not contribute to the survivability of the attack.

3.3. Anti-Ransomware Strategies

Broadly, the strategies employed by existing anti-ransomware can be classified into one of the two categories: preventing the initial execution or detecting an in-progress attack:

3.3.1. Preventing Execution. Ideally, an attack could be stopped before the malware begins executing. This strategy requires the anti-ransomware software to classify a program as malicious or benign before it starts to execute.

One classic mechanism for determining that a provided sample is malicious is to detect a signature that is already known. This requires the maintenance of a database of ransomware signatures (e.g., file hashes or byte strings) to update it as new samples emerge. This approach has the advantage of being reliable, but requires the sample (or a similar sample) to have been previously seen. Similar behavior can be masked behind a dissimilar sample; single byte changes will change a full file hash (e.g., SHA-256) and packing/encryption approaches can prevent byte strings from reappearing in samples. Such variations will evade detection [76].

This technique is useful in scenarios where the sample has stopped working (e.g., because of an unreachable server or a time expiration), if the sample detects sandbox environments [39], or use of stalling code that prevent execution and analysis [44], [53]. Opcode sequence analysis using an n-gram approach is one such method of static analysis [23], [102] used for ransomware detection and classification. Similarly, researchers have used machine learning for feature extraction and detection [80]. A limitation with this is that the binary can be encrypted, password protected, or obfuscated [17], [48], [50], [82], [100], preventing such tools from examining the sample without further work. Further, variants may exhibit their malicious behavior upon runtime only.

A coarse method of protection is to prevent unknown or untrusted processes from accessing the user's data [31]. This requires the user to exercise sound judgment in deciding which programs are malicious or benign and that the filesystem location of their data is known. In practice, the settings to enable ransomware detection are sometimes disabled by default [38]. If the user makes incorrect decisions about the configuration of such a feature, it can be rendered completely ineffective or become unusable as it blocks benign processes and alerts the user. This approach can be defeated by fileless ransomware [62] or process injection into a benign, trusted process [70], [97]. Benign, well-known programs can also be leveraged without modification to perform ransomware attacks [2]. Some commercial anti-ransomware products use this method exclusively, as we will discuss in Section 6.

3.3.2. Behavioral Detection. Once started, the only way to reduce survivability is to quickly detect the attack or to have a recovery system unreliant on detection (e.g., a complete off-site backup). Once detected, the anti-ransomware can execute a file recovery mitigation strategy. Dynamic analysis techniques monitor aspects of the host to determine when a process's behavior has become suspicious [76]:

File Content Changes. Since ransomware manipulates files as a fundamental behavior, a reasonable measurement point for detection is the content of the files being accessed. This includes looking for changes in file parameters like increase in Shannon entropy of a file [18], [44], [46], [48], [85], [88], Kullback-Liebler Divergence [59], similarity measurements [85], rate of change [88], or file structure (as measured by e.g., libmagic) [54], [85].

Filesystem Modification. In addition to measuring the files, some systems monitor aspects of the filesystem for tampering such as the file name (and extension) [22], [34], [64], file size [34], [56], the Windows Master File Table (MFT) [48], [49], and Windows shadow copies [4], [34], [34].

I/O Operations. Real-time hooking and monitoring of I/O requests (between a process and the filesystem) provides a low-level view of the current disk reads and writes [46], [48], [85], [88].

Canary Files. If the ransomware must discover files, then a coarse way to detect an attack would be to place trap files throughout the filesystem, and alert when these are accessed [18], [33], [63]. It remains an open question whether having files that the user cannot access creates usability issues; regardless, a ransomware sample could use a file selection strategy that avoids known or obvious canary files [89].

Cryptography Usage. Detecting the use of specific cryptographic APIs can be useful in detecting when unauthorized software begins a cryptographic operation (e.g., Windows CryptoAPI) [46], [48], [54]. However, a sample could use its own statically-linked library, another internal implementation of cryptography, or an external utility (e.g., GPG [2]).

Network Analysis. After starting execution, the ransomware may connect to a command and control server using a domain generation algorithm or hard-code the address to fetch encryption keys and acquire payment information [7], [39], [54]. Monitoring network communications can aid in the detection of samples that have such behavior [7], [16], [34], [39], [64], [99].

In the next section, we discuss why ransomware detection is difficult and how the above behaviors contribute to detection.

4. The Challenges of Anti-Ransomware

Developing an anti-malware system is difficult: the authors must identify useful features, design and implement a measurement system, and perform a reasonable evaluation. This section examines the two challenges specific to building a ransomware detector:

4.1. Designing for Fundamental Detection

From the definitions in the previous section, we can see that a *robust detector* would detect all variations of fundamental behaviors and have no reliance on general behaviors. Unfortunately, the fundamental behaviors overlap with benign behaviors; at a minimum, ransomware performs actions that a user might perform themselves on their own data. The ransomware detection problem is equivalent, therefore, to the hard problem of determining the *intent* of a sequence of actions. As an example, the Qwerty ransomware attack leveraged a legitimate GPG binary to encrypt the victim's data [2]. From a detection standpoint, this raises the question: *Is GPG ransomware?* We argue that it depends on the intent of the usage. Accordingly, a robust detector, built upon the fundamental behaviors, must not exclude programs from analysis on the basis of identity. Such a detector should, therefore, have the following goals:

- **Real-Time Behavior Analysis**: Static analysis techniques rely on examination of the contents of a particular sample. Attack samples may be packed, encrypted, fileless, or benign. Despite the strong utility of this type of analysis in preventing an attack from starting, the delivery method of the attack is not fundamental. A robust detector, therefore, must analyze the behavior of processes in real-time.
- **Complete Mediation of Data Access**: It must not be possible for data to be accessed without passing through the anti-ransomware system, for example by using a reference monitor [9]. We note that this mediation must extend to all possible mechanisms to access data, including in-memory, via the filesystem, and via raw disk access.
- **Complete Coverage of Fundamental Properties**: The fundamental properties are broad, and there are certainly many ways to implement each. However, a detector should remain successful despite any such variations. We do not claim that complete coverage is possible but rather that increased coverage will correlate with increased detection.
- **Non-Reliance on General Behaviors**: It may be useful to measure general behaviors to aid in faster detection of ransomware. However, if the detector requires a general behavior to detect, it will have false negatives.

Realizing a detector with these properties is challenging. Not all strategies for detection are effective [30], and to the best of our knowledge, no existing system claims to meet or meets all four goals. As shown in Section 3, however, a wide range of systems use general behaviors.

4.2. Evaluation Methodology

Academic works on ransomware detection often report high true positive rates (TPRs). Table 1 shows the evaluation results of several anti-ransomware works which report these values.

The typical methodology for evaluating ransomware detectors consists of acquiring samples, performing the experiments, evicting non-working samples, and analyzing the data. To acquire samples, researchers typically rely on malware collection services (e.g., VirusTotal). These services allow the researcher to search for data elements such as community-provided tags, family and sample names, and hashes. Samples can then be downloaded in bulk. Several papers note that obtaining working samples is difficult [22], [49], [85]; samples acquired in this way are not guaranteed to perform an attack. This may be because the malware is designed to execute only once, it has an expiration time which has passed, its required command and control servers are offline, it detects a virtual machine or sandbox, or it is mislabeled (i.e., it is not ransomware), for example. A common way to evict non-working samples is to perform hashing of the experiments' document corpus before and after the experiment [85]. If the files have been modified in the course of the experiment, the sample is determined to be active. In the literature we surveyed, the failure rate of acquired samples ranges from 12% to 67%, as shown in Table 1.

The experimental analysis is then performed on only working samples. Malware samples can then be grouped into *families* which describe a categorization of samples by behavioral patterns [83]. Individual samples inside the same family may have unique variations of behavior, though similar enough to be classified with its family members. This may lead to bias in the results if families overlap in behavior or the sample-to-family ratio is high. If the ratio is high, systems are trained with many samples that may exhibit similar behavior. Testing on models built from this malware corpus will naturally have high accuracy, since the testing set will exhibit similar behaviors to the training set.

Comparing results between works is not possible due to two issues: First, we observe that authors often perform some pre-processing on the samples found on ransomware

Author	# Samples Collected	# Samples Active	TPR	Working Samples-Per-Family	Method of Collection
Ahmadian et al. [4]	N/A	27	100	N/A	N/A
Ahmed et al. [3]	1,254	673	98.8	48.1	virusshare.com, virustotal.com, crawled malware repositories, online forums
Cabaj et al. [16]	N/A	787	98	393.5	malwr.com, ransomtracker.abuse.ch
Continella et al. [22]	688	383	100	34.8	virustotal.com
Hasan et al. [37]	1,283	360	97	17.1	virusshare.com
Kharaz et al. [46]	3,156	2,121	96.3	176.8	minotauranalysis.com, malwaretips.com, malwareblacklist.com, malware.dontneedcoffee.com
Kharraz et al. [47]	9,432	1,174	100	40.5	minotauranalysis.com, malwaretips.com, malwareblacklist.com, malware.dontneedcoffee.com
Kharraz et al. [49]	3,921	1,359	N/A	90.6	Anibus, Crawling public malware repositories, Manually browsing through security forums
Kolodenker et al. [54]	713	107	79.4	35.7	virustotal.com, malc0de5.com, vxvault.siri-urz.net
Mbol et al. [59]	N/A	N/A	99.95	N/A	N/A
Morato et al. [64]	N/A	54	100	2.8	N/A
Scaife et al. [85]	2,663	492	100	35.1	virustotal.com
Sgandurra et al. [87]	1,450	582	96.3	52.9	virusshare.com
Tseng et al. [93]	N/A	155	N/A	6.7	N/A
Vinayakumar et al. [95]	N/A	755	100	107.9	offensivecomputing.net, contagiodump.blogspot.in, malwr.com, github.com/ytisf/theZoo, virustotal.com, virusshare.com

TABLE 1. RANSOMWARE SAMPLE COLLECTION DATA FROM EXISTING DETECTION LITERATURE

aggregator websites like VirusTotal, and this evaluated malware corpus is not made available. In addition to the obvious implications for reproducibility, the lack of access to these samples also prevents further analysis to identify potential biases in the malware corpora. Second, even if the malware were made available, there is no guarantee that the malware would continue to function in another researcher's experiments. The same issue was highlighted in [12] and [6], where the authors mention that it is difficult to compare different anti-virus/anti-ransomware that did not target the same ransomware families.

4.3. Summary

In the previous section, we outlined the goals of ransomware and the fundamental behaviors that enable the attack to meet these goals. We then examined the current state of anti-ransomware systems and showed that some techniques use general behaviors that are discretionary to the attacker. Detectors which rely on general behaviors will produce false negatives.

This section outlined the two core problems we discovered after reviewing the current state of ransomware detection. We outline the goals that robust anti-ransomware should meet; to the best of our knowledge, no current anti-ransomware work claims to meet or meets these goals. Second, the methodologies commonly used to evaluate the effectiveness of detection systems may cause systems to be overfitted to a small number of behaviors. This issue is not reflective of a problem with the systems being developed, but rather a systemic issue with the difficulty of finding ransomware samples that both reliably perform an attack and exhibit a wide range of behaviors.

We believe that improving the state of testing will inevitably lead to detection systems that target fundamental behaviors, because exercising a wide range of behaviors will naturally begin to exclude general behaviors. In the next section, we design an evaluation framework for anti-ransomware systems.

5. Building an Anti-Ransomware Evaluation Framework

Acquiring working ransomware samples that represent a wide range of behaviors is the single greatest challenge in building detectors. To solve this problem, we propose Farfel, an evaluation framework that allows researchers to improve their development practices and evaluation methodologies. In this section, we design and implement such a reliable, reusable framework.

5.1. Design

The overall goal of our evaluation framework is to allow the researcher to quickly iterate over a wide range of activity. The framework will allow the researcher to select a specific set of behaviors to execute from a wide range of both general and fundamental behaviors. Fundamental behaviors provide a view of the success of a system at detecting ransomware, while general behaviors will clarify if the system relies on discretionary actions. To that end, the framework must follow the following design rules:

First, the framework must enforce semantically correct execution. That is, it must perform exactly one set of behaviors per execution. It also must not be possible to specify more than one mutually exclusive action; for

Category	Behavior Variation	Citation
Data Discovery	Depth-first search	[6], [68], [75], [78], [85]
	Breadth-first search	[64], [65], [85]
	Creates a new process for every subdirectory	[34], [66], [69], [87], [96]
Data Selection	File extension	[1], [24], [33], [51], [55], [68], [77], [85]
	File size	[49], [68], [85]
Input Strategy	Full-file	[24], [29], [33], [41], [46], [54], [85], [87]
	Partial-file	[5], [57], [59], [64], [92]
	Merge several files into one file	[5], [45], [67]
	Execution delay	[24], [40], [42], [44]
Encryption	Symmetric	[21], [49], [55], [72], [77], [91]
	Asymmetric	[8], [21], [61], [68], [73]
Output Strategy	Overwrite the original file	[13], [18], [22], [24], [44], [46], [47], [49], [54], [85]
	Creation of a new file with the same name	[18], [24], [44], [47], [49], [54], [85]
	Change of extension	[1], [8], [20], [21], [24], [25], [61], [73]

TABLE 2. Implemented ransomware behavior variations from existing literature

example, it must not be possible to specify both a depth-first search and a breadth-first search simultaneously. If a hybrid behavior is desired, it must be implemented and selectable individually.

To do this, the framework must also be be extensible. Developing a perfect, complete framework is not possible; as operating systems and application software evolve, new ways of implementing individual actions will likewise do the same. As new behaviors are discovered or conjectured, the framework should grow to accommodate these. To accommodate extensibility, the system must also be modular. It should be possible to add additional behavior variations without needing to modify other variations.

Next, the framework must be self-contained. Behaviors must not require a specific set of conditions on the host unless it also creates those conditions. The framework should be able to execute any selected behaviors without prior configuration of the host where testing will occur. More specifically, the executed behaviors should be deterministic regardless of the host configuration.

Finally, the framework must be safe to handle. The activities performed by the framework are inherently destructive. If executed on a developer's workstation by accident, it must be possible to recover from any damage caused.

5.2. Implementation

We implemented Farfel as a multi-platform Go program, which takes the desired behaviors as input. Upon execution, the code performs the desired actions, then terminates. We created Farfel over seven weeks with two researchers verifying the implementation during development. One person wrote each implementation, and the other developed unit tests to verify the correctness of the code.

To aid the ability for researchers to have confidence in experimental results, if the code encounters any error (including file access errors), it outputs an error to the log and immediately terminates. Conversely, if the code runs to completion, a success message is written to the log. This enables sanity checking of the results to ensure that the results from Farfel match the results from any experiment (e.g., the code should not report success if detected and blocked by anti-ransomware). Any results that do not pass the sanity check can be manually inspected; we will discuss cases where this occurs in Section 6. For safety, when Farfel generates a cryptographic key, it stores it on the local host. We verified that encrypted files are recoverable using the stored key.

Each behavior is implemented independently inside Farfel, such that each input configuration selects a unique execution path through the code. Adding a new behavior subsequently adds a new input configuration and also increases the total number of permutations of behaviors available for testing. To identify candidate behaviors to implement, we reviewed academic literature on ransomware and publicly-available malware reports until we stopped discovering new variations of behaviors; these are shown in Table 2.

Our goal was not to implement a comprehensive, exhaustive list of all possible variations of behaviors; instead, we focused on implementing a range of contrasting behaviors and extensibility. For example, we chose to implement one asymmetric and symmetric algorithm because these are contrasting implementations and hybrid approaches (which are common) are predominantly symmetric on-disk. This still does not account for all possible key lengths, modes, and algorithms. In this case, adding additional encryption behaviors expands our experimental results but does not change those presented in Section 6, which show how our approach can uncover gaps in detectors. Furthermore, Farfel is built in such a way that additional behavior variants can be implemented and added to an evaluation plan at any time.

In total, Farfel consists of 21 unique behavior variations. Figure 1 illustrates how the variations are implemented; mutually-exclusive options are grouped together (as shown in the middle column of the figure). To construct a complete test case, the user is required to select exactly one behavior variation from each of these groups. The test case is then executed in the order shown. After the test case completes, a ransom message is displayed. While not fundamental, the ransom message provides a visual indicator that the process is complete; we note that detecting the ransom message is too late to perform any meaningful blocking. Farfel implements 1,536 unique permutations of behavior variations (test cases), consisting of each of the fundamental behaviors specified in Section 3: Data Discovery (DD), Data Selection (DS), Input Strategy (IS), Encryption (EN), and Output Strategy (OS):

DD-1. Traversal Method. The pattern by which ransomware discovers files specifies the order of I/O calls. This may influence detection if the detector is sensitive to access patterns. We have implemented both a depth-first search (DD-1.1) and a breadth-first search (DD-1.2).

Additionally, we implemented parallelism (DD-1.3): For the current directory, the current process creates two separate lists. One contains the files and the other contains the subdirectories. The files are sent for further processing within the same process, however a new subprocess is created for each subdirectory from the list. These subprocesses recursively continue this process in parallel until reaching maximum depth (in contrast to DD-1.1 and DD-1.2, where all activity is performed in a single process).

DS-1. File Extension. Ransomware often selects specific file extensions to encrypt in order to target more valuable files. DS-1.1 excludes files that do not have an extension of `pdf`, `jpg`, or `doc`. DS-1.2 makes no change to the selected files with respect to the file extension.

DS-2. Small Files. DS-2.1 excludes a file if the file size is less than 5KB. DS-2.2 makes no change to the selected files with respect to the file size.

DS-3. Large Files. DS-3.1 excludes a file if the file size is greater than 1MB. DS-3.2 makes no change to the selected files with respect to the file size.

IS-1. Partial File Input. This variation reads the 25% of the file directly occurring after the midpoint of the data. Intuitively, this may avoid detection in cases where file headers (e.g., using libmagic) are used to identify the structure of a file. IS-1.1 enables this functionality and reads only a portion of the file as described. IS-1.2 reads the full file.

IS-2. Merged File Input. In this variant, whenever the program encounters a directory, it creates a new file named `mergedFile` in that directory, lists all the files in the directory, reads them sequentially, copies the data to the new file and deletes the original files. After the merge is completed, only the `mergedFile` exists in the directory and is passed on for further processing. IS-2.1 enables this functionality and merges the files. IS-2.2 disables the functionality and will output each file separately.

IS-3. Execution Delay. This variation adds a delay to the execution by making the test case sleep for the specified time (in seconds). In cases where the detector uses a time window to measure operations, this delay can obscure the actions that occur within a specified window. IS-3.1 adds a two-second delay between every file encryption while traversing files in the filesystem. IS-3.2 disables the delay and processes files as fast as the host can operate.

EN-1. Algorithm Selection. EN-1.1 enables symmetric encryption using AES; it generates a 256-bit key using the standard Go encryption library. Data is encrypted using the GCM mode of operation. A unique key is generated per test case. The key is saved in a file with the timestamp. EN-1.2 enables asymmetric encryption using RSA; it generates a 2048-bit public and private key pair using the standard Go RSA encryption library. It then saves the public-private key pair in a file. The public-key is used to encrypt the data. For the larger files, the content is broken down in smaller chunks of 126-bytes before encryption. This process in repeated until all the data is encrypted.

OS-1. Change Extension. Ransomware often changes the extension of the file, which has two effects: first, it attempts to indicate to the victim that the files have been compromised. Second, it prevents the file from being opened by the program associated with the original file extension. While this is not a fundamental behavior, we included this to evaluate whether detectors rely on simple general behaviors such as this for detection. OS-1.1 changes the output file extension to `.rsm`. OS-1.2 makes no change to the file's extension.

OS-2. Output File Selection. OS-2.1 deletes the original file then outputs the encrypted data to a new file with the original name. OS-2.2 overwrites the original file with the encrypted data.

We intend to provide our code to researchers upon request and accept contributions to add or modify behaviors. Our hope is that the security community will embrace this framework as an expected methodology for future ransomware works.

6. Evaluation

In order to evaluate the effectiveness of Farfel at identifying gaps in behavioral detection, we executed our tests inside virtual machines running academic and commercial anti-ransomware software. We chose five commercial anti-malware products from a list of highly-rated products from an independent testing firm [10]; we selected products that specifically mention their use in detecting ransomware.

For academic works, we contacted authors of anti-ransomware systems that were real-time ransomware detectors. We attempted to contact 13 anti-ransomware authors but only received a product for two: CryptoDrop [85] and ShieldFS [22]. Early in our work, we attempted to reproduce some academic works but found they did not include enough information to accurately reproduce them (corroborated by [12], [30]). As a result, our attempts to reproduce these would not generate data that accurately represents the original works, and we relied on the detectors we could access. In total, we evaluated seven systems. While the number of evaluated products may seem low

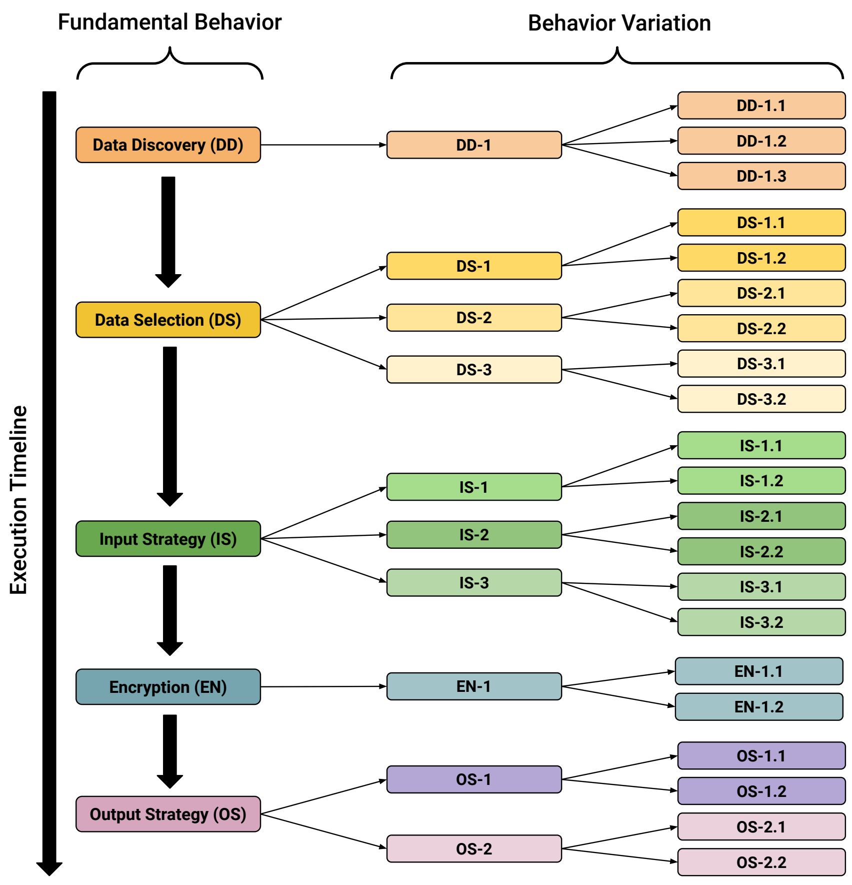


Figure 1. This figure shows our implementation of each of the 21 variations of the fundamental behaviors. To construct a complete test case, the user must select exactly one variation from each group in the center column. These are then executed in the order shown. In total, Farfel implements 1,536 unique permutations of these variations.

relative to the number of anti-ransomware products in existence, we note that our goal is not to evaluate detectors holistically or exhaustively; our evaluation is intended to examine the effectiveness of Farfel in testing anti-ransomware products so that future works can incorporate reproducible tests.

6.1. Experimental Setup

For each anti-ransomware system, we instrumented a virtual machine in QEMU/KVM running Windows 10 with Python 2.7 and the Cuckoo Sandbox agent. Inside each virtual machine, we placed 300 documents inside the user's documents directory. The documents were a subset of the corpus used in [85]. Specifically, we removed any files or directories with a path depth greater than two; this helped to ensure that in cases where the anti-ransomware does not alarm that our experiments do not wait unnecessarily for many thousands of files to be encrypted. The breakdown of our corpus by file extension and count is shown in Table 3.

For speed, the virtual machines were configured with VirtIO controllers and writeback caches for the primary storage. We then installed one anti-ransomware product per guest[2] and verified that the product successfully detects malware using an EICAR test file [26].

2. Windows Defender does not require an installation.

Virtual machines were disconnected from the Internet for several reasons: first, many modern anti-malware products (including those in our experiments) automatically upload unknown binaries or metadata to the vendor for analysis; this cannot always be disabled. When this happens, this creates a severe methodological issue—we cannot guarantee that future results are not tainted by prior knowledge of Farfel's behaviors or identity. Second, while it is true that many ransomware families use a command-and-control server to generate and distribute keys, this is not required [81] and as a result, Internet usage is a general behavior as we described in Section 3. Finally, while the products we tested might access the Internet in ideal circumstances, none of these products indicated during testing that this access was required or that protection is diminished in an offline environment.

Once the guest was configured, we took a snapshot of the guest and cloned the guest to permit experiments to run in parallel. All 1,536 permutations of behavior variants (test cases) were queued for analysis. Cuckoo Sandbox would restore the guest's snapshot, inject the Farfel binary with the experiment's parameters, and periodically check the guest's status. The experiment ran until either our code exited successfully (i.e., all files were encrypted) or the anti-ransomware reported a detection. Farfel's logs were then extracted from the guest. The guest was then halted,

Extension	File Count	Median File Size (Bytes)
jpg	51	90267
pdf	46	133473
html	29	21535
xls	26	42752
xml	23	3451
doc	21	78336
gif	15	10222
ppt	12	871424
txt	10	23555
gz	8	683245
csv	6	14451
mov	6	3104758
unk	5	20160
log	5	59288
pptx	4	1184586
java	3	3799
docx	3	30318
mp3	3	267648
dbase3	2	1874
css	2	7576
js	2	18827
eps	2	31798
xlsx	2	46400
m4a	2	453441
png	2	650369
text	1	413
hlp	1	1361
kml	1	514
kmz	1	1596
sys	1	1823
rtf	1	6891
sql	1	5841
zip	1	77430
ps	1	257224
tif	1	9807680

TABLE 3. COUNT AND MEDIAN SIZE OF THE FILES (BYTES) PER EXTENSION IN THE EXPERIMENTAL CORPUS

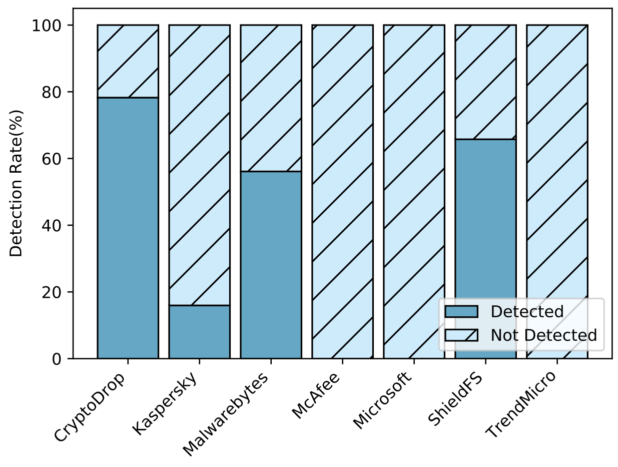


Figure 2. This figure shows the detection rate for each of the seven anti-ransomware products we tested against our 1,536 test cases. Among academic works, CryptoDrop achieved the highest detection rate at 78%; among the commercial packages, Malwarebytes at 56%. Three of the commercial products we tested did not detect any of our test cases.

restored from the clean snapshot, and another test case was started.

We validated our experimental data to ensure that reported conditions by Farfel matched the anti-ransomware. For example, if it reported success, the anti-ransomware should not have alerted. Conversely, in the case of a detection, it should not complete successfully. In the cases where our check did not pass, we investigated each manually. We discuss each of these cases below as part of the individual experimental results.

6.2. Results

The aggregate results from each of the experiments is presented in Figure 2. We considered that any detection logged by the anti-ransomware was a success for the product. Similarly, in cases where multiple processes were spawned (DD-1.3), detection of any process constituted total success. We chose this methodology since once any detection occurs, a separate mitigation activity could continue cleaning the host as a matter of policy (e.g., taking the host offline, killing parent processes, or restoring files).

No anti-ransomware we evaluated with Farfel detected all of the test cases. Three products detected in zero cases; we will discuss these in further detail individually below. Figure 3 shows a matrix generated from a pair-wise correlation between individual variations and detection in the four products that exhibited any alerts; this data identifies specific gaps in behavioral detection coverage. Equal and opposite correlation is shown between mutually exclusive variations as described in Section 5.

CryptoDrop Anti-Ransomware 1.5. CryptoDrop had the highest number of detections (1,202, 78%) among all products we tested. Upon a deeper analysis, we discovered a strong negative correlation between DD-1.3 (parallelism) and detection; out of the 512 cases with DD-1.3 selected, 309 of them go undetected. Similarly, we also discovered a negative correlation with IS-2.1 (merging); out of 768 cases with IS-2.1 selected, 256 go undetected. All of the undetected IS-2.1 cases also had DD-1.3 selected. CryptoDrop detected no cases where both IS-2.1 and DD-1.3 were selected, which implies that CryptoDrop may have a coverage gap when subprocesses are created or when the input file is manipulated prior to encryption. CryptoDrop does not appear to have any other strong negative correlations with our behavior variations.

In [85], a scoring system for processes is described, such that the product alerts when a process achieves a sufficient score. In cases where a small number of actions are performed in a single process (e.g., in DD-1.3), a sufficient score is not generated for any one process. Furthermore, because CryptoDrop pre-measures files for later analysis, IS-2.1 is problematic for this product due to the encryption of a new file (i.e., `mergedFile`) rather than existing data.

Kaspersky Anti-Virus 20.0.14.1085(k). This product detected 245 test cases (16%). Our analysis revealed strong negative correlations for both IS-1.1 (partial file) and IS-2.1 (merging). In particular, Kaspersky detected no cases where either IS-1.1, IS-2.1, or both are selected. This accounted for 1,152 undetected cases (89% of total undetected). While we do not have access to the specifics of how this product implements detection, it appears to be sensitive to behavior variations related to input strategy. The remaining undetected cases did not have a distinct correlation with any one variation.

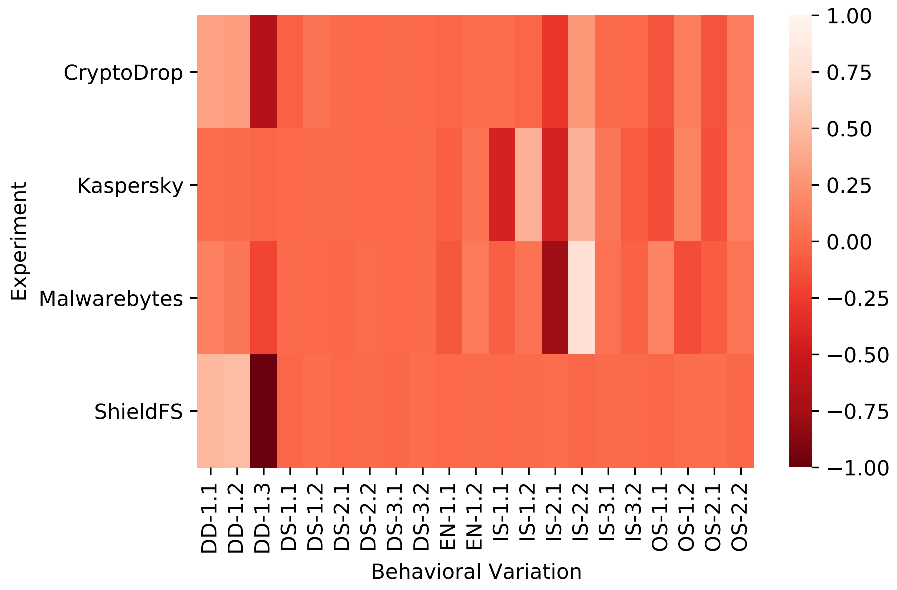


Figure 3. This figure shows the pair-wise correlation between individual behavior variations and detection in our experiments. The lighter shades in the figure represent a positive correlation with detection while the darker shades represent a negative correlation with detection. Strong negative correlations indicate gaps in behavioral coverage by the detectors. Three products are not shown in this figure because they detected zero test cases.

Malwarebytes 4.1.0. This product detected the most cases of the commercial products, detecting 862 test cases (56%). The strongest negative correlation among behavior variations was IS-2.1 (merging), constituting 93% of the undetected cases. IS-2.1 is not a complete failure for this product, however, because 140 of cases with this variation enabled were detected. Of these, 132 cases had OS-1.1 (file extension) selected. We found no other strong correlations with single behavior variants.

In 13 cases, we observed anomalies in our data validation that caused us to investigate further. In one of the test cases, Malwarebytes deleted our binary upon detection but before the detection log was written, causing the logs to appear as if there was no detection and the test case did not complete. In the 12 remaining cases, Farfel had completed encryption of all of the user's files (denoted by the creation of success message to the log), however the software had also successfully detected the presence of ransomware. To clarify the sequence of events, we examined the screenshots taken in each virtual machine. From this, we noted that in each of these cases, the "detected" alert from Malwarebytes appeared after our test case completed encryption of all files and was in the process of displaying the ransom message. In all 13 cases, we manually revised our data to reflect successful detection.

McAfee Total Protection 16.0. This product detected no test cases, however the product's marketing indicates ransomware protection [60]. We re-verified that the product was enabled and successfully detected the EICAR test file, but we were unable to cause any detections with our test cases. We found no ransomware-specific options in the product. It is possible that this product does not implement behavioral detection or that it does not function when not connected to the Internet.

Microsoft Windows Defender 4.18.2004.6. This product detected no test cases. Similar to McAfee, we verified that Windows Defender was enabled and successfully detected a test file. This product has an anti-ransomware access control feature that blocks processes not explicitly permitted by the user. Since our test cases are implemented in a single binary which is unknown to the products, enabling this access control feature would deny our binary from executing and prevent our analysis. Similarly, enabling the feature and permitting our binary would effectively bypass the access control; we left this feature disabled. To the best of our knowledge, enabling this feature activates no other functionality. We discuss the implications of access control features such as the one provided in this product in Section 7.

ShieldFS. The authors of [22] provided us with a Windows 10 virtual machine where the ShieldFS software was already installed and configured to run at boot. We did not have access to the source code; the virtual machine was instrumented with pre-compiled binaries. Instead of attempting to transplant the ShieldFS software into a new guest, we copied our file corpus and the Farfel binary into the provided guest. In our initial tests while verifying the functionality of our code, we noticed that ShieldFS failed to detect any test cases. We also noticed that in [22] the authors describe that the detector becomes more sensitive as the percentage of files (out of the total number of the user's files) accessed by a particular process increases. Since our corpus is two orders of magnitude smaller than the total number of files in the provided virtual machine, we tested again after removing the files provided in the machine and ShieldFS began to successfully detect our test cases.

In total, ShieldFS detected 1,010 test cases (65.7%). Our analysis uncovered a strong negative correlation with DD-1.3 (parallelism) which contributed to 512 (97%) of the undetected cases. ShieldFS detected no cases where this behavior variation was selected. We suspect that, like CryptoDrop, the small amount of activity per process influences this detector's ability to alert.

There was a single anomaly in our results when ShieldFS logged a detection but our test case completed successfully. Upon manual inspection, we observed that ShieldFS terminates a detected process after a delay. In this case, the delay was such that our test case had time to complete after detection. In keeping with our methodology of labelling success for the detector after any detection, we manually revised our data to reflect success in this case.

Trend Micro Maximum Security 16.0.1277. Similar to both McAfee and Windows Defender, this product detected zero test cases. We again verified that the product was enabled and successfully detected a test file. Trend Micro includes an access control feature (like Windows Defender) called Folder Shield which allows users to explicitly permit programs which can access protected directories. To the best of our knowledge, this feature has no other functionality. As with Windows Defender, we left this feature disabled since enabling it would prevent our experiments from executing.

6.3. Summary

Of the seven products we evaluated, no product detected all of our test cases. Three commercial products detected zero cases. In the products that had successful detections, we observe that the behavior coverage gap in all of these is related to DD-1.3, IS-1.1, and/or IS-2.1. These three behavior variations occur *prior to* encryption, which provides two key insights into gaps in existing behavioral coverage: First, actions which occur directly and completely on the original data appear to trigger detection more reliably than if the input file is manipulated prior to encryption. Second, the two academic works we evaluated have gaps with multi-process behavior; future work should consider how to detect in cases where multiple processes are used.

In general, this evaluation demonstrates that testing anti-ransomware software with a wide range of behavior variations can uncover coverage gaps in detectors. Our test cases were complete attacks and even single changes to selected variations can result in non-detection. Accordingly, we show that it is trivial to bypass existing ransomware detectors and that more robust testing is needed. In the next section, we discuss the implications of the access control mechanisms we discovered and of other aspects of testing.

7. Discussion

7.1. Limitations

Farfel is not replacement for existing development or testing methods such as using live malware samples. Rather, we view this as a critical supplement for identifying a detector's weak spots. As we mentioned earlier, we suspect having full coverage of all possible fundamental behavior variations is not possible due to the overlap between benign and malicious behaviors and the specific mechanisms used by ransomware will undoubtedly change over time. Since the test cases consist only of valid ransomware behaviors, any negative detection result is false; detectors evaluated with Farfel should aim for 100% detection of test cases. A secondary metric is the speed at which detection occurs. While this could be measured in units of time, we believe that measurement by amount of data lost [85] is a more realistic measure of efficacy.

Complete behavioral coverage is unlikely. Farfel's coverage can be measured over time as a total number of permutations of behavior variations compared to a previous state. New behavior variations will be sourced from both new malware samples and researchers' models of attack. The former is a lagging source of behaviors, but we expect that as the number of unique tests increases, the likelihood of a new sample that is not partially or completely covered by existing tests will decrease. A useful approach to identifying new variations could perform formal modeling of disk I/O in Windows, for example. Such a model could more completely enumerate the methods for accessing stored data, which could then be used as new variants and test cases. In our experience with developing ransomware detectors, addressing a broader category of behaviors (e.g., parallelism) will add coverage for a wide range of possible variations.

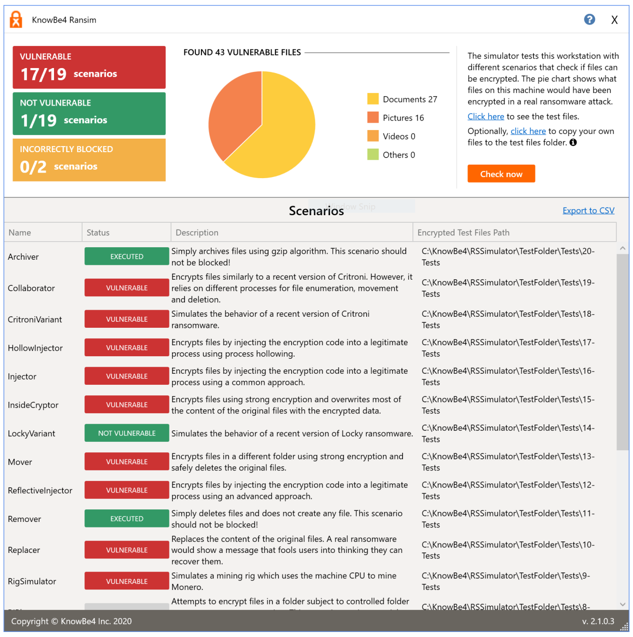
7.2. Access Control

As we discussed in Section 6, two commercial products we evaluated offer a strict access control mechanism. The motivation for such a strong control is that processes accessing the user's data should be explicitly permitted by the user. We believe that this is a reasonable and effective security control when the identity of the binary is unknown. However, access control is ineffective if the process has already been permitted by the user or is permitted by default. In such cases, a compromised or misused process will bypass the control; we discussed this in Section 4. We expect that strong access control coupled with a behavioral detector will improve coverage of attacks, as the detector will only need to monitor permitted processes.

The primary disadvantage to using both access control and behavioral detection is that the user may be asked multiple times to approve the same program. For example, a program may need to be approved to gain access to the user's files and then again if it exhibits fundamental behaviors on that data. This may result in alert fatigue or confusion for the user, however, we leave this problem to future work.

7.3. False Positive Evaluation

Among academic works, testing for false positives using benign programs (e.g., 7-Zip) is common [22], [85]. Where these tests are performed, the detector is evaluated as correct if it does not alert on the activity of a benign program. Based on our description in Section 4, such an evaluation is flawed since a benign program may be leveraged to execute a ransomware attack. This could cause a system to be trained to produce false negatives when



VULNERABLE
17/19 scenarios

NOT VULNERABLE
1/19 scenarios

INCORRECTLY BLOCKED
0/2 scenarios

The simulator tests this workstation with different scenarios that check if files can be encrypted. The pie chart shows what files on this machine would have been encrypted in a real ransomware attack.

Click here to see the test files.

Optionally, click here to copy your own files to the test files folder.

Scenarios

Name	Status	Description	Encrypted Test Files Path
Archiver	EXECUTED	Simply archives files using gzip algorithm. This scenario should not be blocked!	C:\KnowBe4\RSSimulator\TestFolder\Tests\20-Tests
Collaborator	VULNERABLE	Encrypts files similarly to a recent version of Critroni. However, it relies on different processes for file enumeration, movement and deletion.	C:\KnowBe4\RSSimulator\TestFolder\Tests\19-Tests
CritroniVariant	VULNERABLE	Simulates the behavior of a recent version of Critroni ransomware.	C:\KnowBe4\RSSimulator\TestFolder\Tests\18-Tests
HollowInjector	VULNERABLE	Encrypts files by injecting the encryption code into a legitimate process using process hollowing.	C:\KnowBe4\RSSimulator\TestFolder\Tests\17-Tests
Injector	VULNERABLE	Encrypts files by injecting the encryption code into a legitimate process using a common approach.	C:\KnowBe4\RSSimulator\TestFolder\Tests\16-Tests
InsideCryptor	VULNERABLE	Encrypts files using strong encryption and overwrites most of the content of the original files with the encrypted data.	C:\KnowBe4\RSSimulator\TestFolder\Tests\15-Tests
LockyVariant	NOT VULNERABLE	Simulates the behavior of a recent version of Locky ransomware.	C:\KnowBe4\RSSimulator\TestFolder\Tests\14-Tests
Mover	VULNERABLE	Encrypts files in a different folder using strong encryption and safely deletes the original files.	C:\KnowBe4\RSSimulator\TestFolder\Tests\13-Tests
ReflectiveInjector	VULNERABLE	Encrypts files by injecting the encryption code into a legitimate process using an advanced approach.	C:\KnowBe4\RSSimulator\TestFolder\Tests\12-Tests
Remover	EXECUTED	Simply deletes files and does not create any file. This scenario should not be blocked!	C:\KnowBe4\RSSimulator\TestFolder\Tests\11-Tests
Replacer	VULNERABLE	Replaces the content of the original files. A real ransomware would show a message that fools users into thinking they can recover them.	C:\KnowBe4\RSSimulator\TestFolder\Tests\10-Tests
RigSimulator	VULNERABLE	Simulates a mining rig which uses the machine CPU to mine Monero.	C:\KnowBe4\RSSimulator\TestFolder\Tests\9-Tests
		Attempts to encrypt files in a folder subject to controlled folder	C:\KnowBe4\RSSimulator\TestFolder\Tests\8-

Copyright © KnowBe4 Inc. 2020 v. 2.1.0.3

Figure 4. This is a screenshot of RanSim, a commercial ransomware simulator. While this simulator can be useful for consumers, it does not employ a valid methodology for scientific research.

benign programs become malicious. A better approach to false positive testing would be to evaluate whether a detection occurred *without* the presence of fundamental behaviors. This new definition implicitly accepts detection on benign programs if the program exhibits fundamental behaviors.

We propose that future work in ransomware detection would benefit from studies in new ways of evaluating false positive measurement for anti-ransomware. If we cannot rely on fundamental behaviors to differentiate benign and malicious software, we will require new heuristics to reduce errors. For example, Fukushima et al. observe behaviors that are common for some types of programs (e.g., installers) and not for others [28]. While this method will not strictly decide whether a program is ransomware, if a detector has already identified fundamental behavior, such a heuristic could be used to decide whether (and which) action to take. The workflow of a detector should detect fundamentals *first* then employ other heuristics to suppress alarms; performing these operations in reverse leaves the possibility for false positives since a program is decided to be benign prior to observation of fundamental behaviors.

From a systems perspective, measuring how users exercise programs to exhibit fundamental behaviors may identify new paths for context-based detection. For example, if a user has engaged with a program recently and has never used a program on a specific type of file (e.g., a word processor begins to modify video files), it may be useful to alert. Second, an evaluation of how such alerts (i.e., "Is this you making this change?") are perceived by users could provide insights into how these systems can avoid frequently interrupting the user.

7.4. Ransomware Simulation

A commercial ransomware simulator, RanSim [52] (shown in Figure 4), executes scenarios with the intent of providing the user insight into their own system's vulnerability to ransomware. The version of software we examined contains 21 total scenarios: 18 ransomware scenarios, 2 benign scenarios, and 1 cryptomining scenario (using XmRig). RanSim is closed source and is distributed as a set of heavily obfuscated .NET/Mono assemblies. While the product's documentation provides some insight into the behavior of each scenario (e.g., "Simulates the behavior of a recent version of Thor ransomware."), to evaluate its suitability for scientific research, we instrumented a host running the software to identify fundamental behaviors by recording the operating system activity during execution. For our analysis, we exclude the cryptomining scenario, which solely unpacks a cryptocurrency mining program and executes it, because it is not ransomware (and therefore out of our scope). Due to the opaqueness of cryptographic operations to our Windows instrumentation, we were unable to analyze the algorithm selection and other cryptographic parameters for each scenario, however we did observe the software accessing libraries and registry keys that would indicate the use of built-in cryptographic libraries.

RanSim unpacks a directory for each scenario containing 29 documents (6 of these are duplicates of each other) including CSV, JPG, DOCX, PDF, PNG, PPTX, and XLSX. All of the victim files are contained in that single directory; there are no subdirectories. The product allows the user to add their own files to the directory, however we were unable to cause any scenarios to access files in subdirectories when provided. When the user begins the analysis, all scenarios are executed in parallel. If a scenario completes successfully, the user interface marks that the current host is "vulnerable."

Behaviors. We discovered that each of the scenarios queries the directory to list its files, then performs its operation on them in lexical order (a single DD behavior). No scenario appeared to select a subset of files for operation (a single DS behavior). We found the greatest variety of fundamental behavior variation in output strategy, despite overlap between scenarios. Over half of the scenarios differ only in whether they delete or rename the original file, when they perform that operation, and whether or not they overwrite the original file before doing so. Three scenarios inject malicious code into Notepad, which then performs the file operations and encryption, but we observed the same fundamental behaviors once Notepad was compromised. The first scenario only checks whether access controls prevent unknown processes from accessing the files, then exits. The second scenario pre- and post-processes files by spawning a `cmd.exe` process to copy the original file and delete it, but uses the base scenario executable for reading and writing content. Finally, the third scenario is described as simulating a network connection to send a key to the server (a general behavior), but we did not observe this behavior during execution. In summary, we found that RanSim covers variety of output strategies, but it does not exercise a wide range of fundamental behaviors.

Benign Scenarios. RanSim contains two scenarios which it notes should not be blocked, with the implication that these are false positives if detected. The first simply iterates and deletes the files; since this does not meet the goals of survivability and reversibility as presented in Section 3, this correctly does not meet the requirements for a ransomware attack. However, the second scenario reads an original file, compresses the data, writes the compressed data into a new file, then deletes the original file. This scenario is functionally the same as several of the other scenarios (i.e., read, write new file, delete) except that it processes the data using a compression algorithm instead of an encryption algorithm. Extending from our earlier discussion of false positives, we believe future research would benefit from exploring whether it is possible to exclude this scenario from a robust ransomware detector given the difficulty of distinguishing these outputs [35].

Suitability. This product is not designed for scientific research. For this use, it exhibits critical methodological issues. First, it executes all tests in parallel; this does not allow for resetting the environment between each scenario which will manipulate detectors such as CryptoDrop and ShieldFS where detection is either based on activity-over-time or number of files modified. Second, it encrypts only within a test directory, which by default is outside of standard monitored directories for user data. Third, it is a relatively well-known product to anti-malware vendors; although the product avoids simple hash detection by modifying 4 bytes in each scenario's binary, the remainder of the binaries are identical. At the time of writing, 26 malware engines detect the scenario binaries as malicious on VirusTotal, with one (Malwarebytes) specifically detecting the binary as `RiskWare.RansomSimulator`. In cases where the binary is detected by its identity, behavior detection is not needed or used, and the ability to detect the simulation may cause vendors to manipulate their products' behaviors in its presence. Finally, RanSim's tests include behaviors which do not provide insight into specific coverage gaps. For example, three of the cases from RanSim note it "simulates the behavior of a recent version of Locky/Critroni/Thor ransomware." Locky alone has multiple variants per month [81], making it unclear which specific behaviors are implemented.

This product differs from Farfel in several key ways: Farfel currently contains nearly two orders of magnitude more test cases than RanSim. It also allows developers and scientists to individually select behavior variations to test, allowing granularity and control over experiments and providing valuable feedback about gaps in detection coverage. Our testing methodology resets the testbed after each test case, preventing detectors from being influenced by previous cases. Finally, Farfel is extensible by our community to add new behaviors as they emerge. Ultimately, RanSim is designed for consumers to identify if their anti-ransomware software is operating and not to identify specific failure points.

7.5. Future Directions and Summary

We believe that the future of ransomware mitigation is not classifying programs as benign or malicious. As we have discussed throughout this work, a robust detector must identify sequences of actions that are fundamental to this attack, despite a program's identity. Towards the development of such detectors, we expect that future work will take three directions: implementing new behavior variations, building new heuristics for approximating intent, and increasing the speed of testing. We discussed the first two points earlier in this section.

A single set of our experiments for a single anti-malware product took approximately one week to execute. As new behavior variations are implemented, the number of test cases rises exponentially. We suspect that one valuable contribution would be to develop a scalable (perhaps cloud-based) testing platform for researchers to deploy detectors onto. Since tests can be executed in parallel, speed could be increased while keeping similar costs. Existing platforms, such as Cuckoo Sandbox as used in this work and others, require custom deployments and creates a substantial burden to running experiments. Solving this issue would lower the friction of developing and testing new anti-ransomware software.

8. Conclusion

Ransomware continues to increase in both prevalence and severity; attackers rely on strong cryptography to ensure survivability and reversibility of their attacks. The success of these attacks have spawned both academic and commercial solutions for attempting to detect ransomware. In this paper, we examined the challenges of developing robust anti-ransomware. First, we characterized the fundamental behaviors of ransomware and common anti-ransomware strategies. Next, we outlined the two major practical challenges of developing robust anti-ransomware: insufficient testing means detectors are not evaluated against a wide range of behaviors. Finally, we design and implement Farfel, an extensible framework for testing anti-ransomware software. We evaluated seven anti-ransomware products with our framework and discovered that no product had complete detection; this feedback is valuable to detector developers for better covering behaviors exhibited by ransomware. We believe that improved testing will result in improved detectors, and we intend to provide Farfel to researchers upon request to allow the community to build new behavior variations.

References

[1] Lawrence Abrams. Researcher finds the karma ransomware being distributed via pay-per-install network. https://www.bleepingcomputer.com/news/security/researcher-finds-the-karma-ransomware-being-distributed-via-pay-per-install-network/, Nov 2016.

[2] Lawrence Abrams. Qwerty ransomware utilizes GnuPG to encrypt a victims files. https://www.bleepingcomputer.com/news/security/qwerty-ransomware-utilizes-gnupg-to-encrypt-a-victims-files/, Mar 2018.

[3] Yahye Abukar, Barış Koçer, and Bander Al-rimy. Automated analysis approach for the detection of high survivable ransomware. *KSII Transactions on Internet and Information Systems*, 14:2236, May 2020.

[4] Mohammad Mehdi Ahmadian and Hamid Reza Shahriari. 2entfox: A framework for high survivable ransomwares detection. In *13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pages 79–84. IEEE, 2016.

[5] Anand Ajjan. Ransomware: Next-generation fake antivirus. *A SophosLabs technical paper*, 2013.

[6] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security*, 74:144–166, 2018.

[7] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE Access*, 7:47053–47067, 2019.

[8] Malware Intelligence Analyst. Cerber ransomware: new, but mature. https://blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/, Mar 2018.

[9] James P Anderson. Computer security technology planning study. Technical report, 1972.

[10] AV-Test. Test antivirus software for windows 10 - february 2020. https://www.av-test.org/en/antivirus/home-windows/windows-10/february-2020/, Mar 2020.

[11] Brian Barrett. The garmin hack was a warning. https://www.wired.com/story/garmin-ransomware-hack-warning/, Aug 2020.

[12] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. A survey on detection techniques for cryptographic ransomware. *IEEE Access*, 7:144925–144944, 2019.

[13] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. Open repository for the evaluation of ransomware detection tools. *IEEE Access*, 8:65658–65669, 2020.

[14] Elisa Bertino and Nayeem Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[15] Christopher Bing and Joseph Menn. Building wave of ransomware attacks strike u.s. hospitals. https://www.reuters.com/article/us-usa-healthcare-cyber/building-wave-of-ransomware-attacks-strike-u-s-hospitals-idUSKBN27D35U, Oct 2020.

[16] Krzysztof Cabaj, Marcin Gregorczyk, and Wojciech Mazurczyk. Software-defined networking-based crypto ransomware detection using http traffic characteristics. *Computers & Electrical Engineering*, 66:353–368, 2018.

[17] J Calvet. Cryptographic function identification in obfuscated binary programs. In *Reverse Engineering Conference*, 2012.

[18] JB Christensen and Niels Beuschau. *Ransomware detection and mitigation tool*. PhD thesis, M. Sc. Thesis, Technical University of Denmark, 2017. https://www2.imm.dtu.dk/pubdb/edoc/imm7039.pdf.

[19] Mihai Christodorescu and Somesh Jha. Testing malware detectors. *ACM SIGSOFT Software Engineering Notes*, 29(4):34–44, 2004.

[20] Catalin Cimpanu. Marlboro ransomware defeated in one day. https://www.bleepingcomputer.com/news/security/marlboro-ransomware-defeated-in-one-day/, Jan 2017.

[21] Aviad Cohen and Nir Nissim. Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. *Expert Systems with Applications*, 102:158–178, 2018.

[22] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347, 2016.

[23] KAO Da-Yu, Shou-Ching HSIAO, and TSO Raylin. Analyzing wannacry ransomware considering the weapons and exploits. In *21st International Conference on Advanced Communication Technology (ICACT)*, pages 1098–1107. IEEE, 2019.

[24] Tooska Dargahi, Ali Dehghantanha, Pooneh Nikkhah Bahrami, Mauro Conti, Giuseppe Bianchi, and Loris Benedetto. A cyber-kill-chain based taxonomy of crypto-ransomware features. *Journal of Computer Virology and Hacking Techniques*, 15(4):277–305, 2019.

[25] Paul Ducklin. "Locky" ransomware – what you need to know. https://nakedsecurity.sophos.com/2016/02/17/locky-ransomware-what-you-need-to-know/, Feb 2016.

[26] European Institute for Computer Anti-Virus Research. Download Anti Malware Testfile. https://www.eicar.org/?page_id=3950.

[27] Jim Finkle. Ransomware: Extortionist hackers borrow customer-service tactics. https://www.reuters.com/article/us-usa-cyber-ransomware-idUSKCN0X917X, Apr 2016.

[28] Yoshiro Fukushima, Akihiro Sakai, Yoshiaki Hori, and Kouichi Sakurai. A behavior based malware detection scheme for avoiding false positive. In *2010 6th IEEE workshop on secure network protocols*, pages 79–84. IEEE, 2010.

[29] Ziya Alper Genç, Gabriele Lenzini, and Peter Ryan. The cipher, the random and the ransom: A survey on current and future ransomware. *Advances in Cybersecurity*, 2017.

[30] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. Next generation cryptographic ransomware. In *Nordic Conference on Secure IT Systems*, pages 385–401. Springer, 2018.

[31] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. No random, no ransom: a key to stop cryptographic ransomware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 234–255. Springer, 2018.

[32] Jurijs Girtakovskis, Ken Jacobi, David Kennerley, Kiran Kumar, Grayson Milbourne, Tyler Moffitt, Cameron Palan, and Steve Snyder. 2017 webroot threat report. https://webroot-cms-cdn.s3.amazonaws.com/8114/8667/3161/Webroot_2017_Threat_Report_US.pdf, 2017.

[33] JA Gómez-Hernández, L Álvarez-González, and Pedro García-Teodoro. R-locker: Thwarting ransomware action through a honeyfile-based approach. *Computers & Security*, 73:389–398, 2018.

[34] Parth S Goyal, Akshat Kakkar, Gopika Vinod, and Gigi Joseph. Crypto-ransomware detection using behavioural analysis. In *Reliability, Safety and Hazard Assessment for Risk-Based Technologies*, pages 239–251. Springer, 2020.

[35] Daniel Hahn, Noah Apthorpe, and Nick Feamster. Detecting compressed cleartext traffic from consumer internet of things devices. *arXiv preprint arXiv:1805.02722*, 2018.

[36] Jaehyun Han, Zhiqiang Lin, and Donald E. Porter. On the effectiveness ofbehavior-based ransomware detection. *EAI SecureComm*, 2020.

[37] Md Mahbub Hasan and Md Mahbubur Rahman. Ranshunt: A support vector machines based ransomware analysis framework with integrated feature set. In *20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–7. IEEE, 2017.

[38] Brendan Hesse. Why you should use windows defender's ransomware prevention. https://lifehacker.com/why-you-should-use-windows-defenders-ransomware-prevent-1837311176, Aug 2019.

[39] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *IEEE Symposium on Security and Privacy (SP)*, pages 618–631, 2018.

[40] Gavin Hull, Henna John, and Budi Arief. Ransomware deployment methods and analysis: views from a predictive model and human responses. *Crime Science*, 8(1):2, 2019.

[41] Blaine M Jeffries. *Securing Critical Infrastructure: A Ransomware Study*. PhD thesis, 2018. https://apps.dtic.mil/dtic/tr/fulltext/u2/1056153.pdf.

[42] Mark Johnson. The 2018 ransomware trends that will keep you up at night. https://info.arcserve.com/blog/insights/2018-ransomware-trends, Apr 2018.

[43] O'Ryan Johnson. Cognizant contains maze ransomware attack as cleanup costs spiral. https://www.crn.com/news/channel-programs/cognizant-contains-maze-ransomware-attack-as-cleanup-costs-spiral, May 2020.

[44] Ashwini Balkrushna Kardile et al. *Crypto Ransomware Analysis and Detection Using Process Monitor*. PhD thesis, 2017. https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/27184/KARDILE-THESIS-2017.pdf?sequence=1.

[45] Dawn Kawamoto. Trojan cryzip extorts decryption fee. https://www.zdnet.com/article/trojan-cryzip-extorts-decryption-fee/, Mar 2006.

[46] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. UNVEIL: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772, 2016.

[47] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 98–119. Springer, 2017.

[48] Amin Kharraz and William Robertson. *Techniques and Solutions for Addressing Ransomware Attacks*. PhD thesis, Northeastern University, Boston, MA, USA., 2017. https://www.ccs.neu.edu/home/mkharraz/publications/amin_kharraz_proposal.pdf.

[49] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[50] Amin Kharraz, William Robertson, and Engin Kirda. Protecting against ransomware: A new line of research or restating classic ideas? *IEEE Security & Privacy*, 16(3):103–107, 2018.

[51] Linas Kiguolis. Remove globeimposter 2.0 ransomware / virus (removal guide) - updated mar 2019. https://www.2-spyware.com/remove-globeimposter-2-0-ransomware-virus.html, Mar 2019.

[52] KnowBe4. Ransomware simulator. https://www.knowbe4.com/ransomware-simulator.

[53] Clemens Kolbitsch, Engin Kirda, and Christopher Kruegel. The power of procrastination: detection and mitigation of execution-stalling malicious code. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 285–296, 2011.

[54] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. Paybreak: Defense against cryptographic ransomware. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security*, pages 599–611, 2017.

[55] Malwarebytes Labs. Explained: Spora ransomware. https://blog.malwarebytes.com/threat-analysis/2017/03/spora-ransomware/, Mar 2017.

[56] Ben Lelonek and Nate Rogers. Exploring some of the many uses of event tracing for windows (etw). https://ruxcon.org.au/assets/2016/slides/ETW_16_RUXCON_NJR_no_notes.pdf, 2016.

[57] Mark Loman. How ransomware attacks. https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-ransomware-behavior-report.pdf, Nov 2019.

[58] Peter Mackenzie and Tilly Travers. The realities of ransomware: A victim's-eye view of an attack. https://news.sophos.com/en-us/2020/08/04/the-realities-of-ransomware-a-victims-eye-view-of-an-attack/, Sep 2020.

[59] Faustin Mbol, Jean-Marc Robert, and Alireza Sadighian. An efficient approach to detect torrentlocker ransomware in computer systems. In *International Conference on Cryptology and Network Security*, pages 532–541. Springer, 2016.

[60] McAfee. McAfee total protection 2020: Antivirus software. https://web.archive.org/web/20200618203136/https://www.mcafee.com/en-us/antivirus/mcafee-total-protection.html, Jun 2020.

[61] Tomas Meskauskas. Cryptxxx ransomware removal instructions. https://www.pcrisk.com/removal-guides/9963-cryptxxx-ransomware, Sep 2016.

[62] Trend Micro. Netwalker fileless ransomware injected via reflective loading. https://blog.trendmicro.com/trendlabs-security-intelligence/netwalker-fileless-ransomware-injected-via-reflective-loading/, Jun 2020.

[63] Chris Moore. Detecting ransomware with honeypot techniques. In *Cybersecurity and Cyberforensics Conference (CCC)*, pages 77–81. IEEE, 2016.

[64] Daniel Morato, Eduardo Berrueta, Eduardo Magaña, and Mikel Izal. Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, 124, 09 2018.

[65] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet. Ransomware's early mitigation mechanisms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.

[66] Ryan Murphy. How does ransomware work? https://www.carbonblack.com/2017/05/12/how-does-ransomware-work/, May 2017.

[67] Ibrahim Nadir and Taimur Bakhshi. Contemporary cybercrime: A taxonomy of ransomware threats & mitigation techniques. In *International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–7. IEEE, 2018.

[68] Pranav Narain. *Ransomware-Rising Menace to an Unsuspecting Cyber Audience*. PhD thesis, 2018. https://uh-ir.tdl.org/bitstream/handle/10657/3145/NARAIN-THESIS-2018.pdf?sequence=1.

[69] Daniel Nieuwenhuizen. A behavioural-based approach to ransomware detection. *Whitepaper. MWR Labs Whitepaper*, 2017.

[70] Cybereason Nocturnus. Sodinokibi: The crown prince of ransomware. https://www.cybereason.com/blog/the-sodinokibi-ransomware-attack, Aug 2019.

[71] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. In *USENIX Security Symposium*, pages 91–106, 2008.

[72] Aurélien Palisse, Hélène Le Bouder, Jean-Louis Lanet, Colas Le Guernic, and Axel Legay. Ransomware and the legacy crypto api. In *International Conference on Risks and Security of Internet and Systems*, pages 11–28. Springer, 2016.

[73] Dorka Palotay. Ransomware in action. https://www.tmsi.hu/antidotum-prezentaciok/antidotum-2017-r1/Ransomware_presentation_DP.pdf, Mar 2017.

[74] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[75] Simon Parkinson, Andrew Crampton, and Richard Hill. *Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach*. Springer, 2018.

[76] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[77] Ashish Patel and Jinal Tailor. A malicious activity monitoring mechanism to detect and prevent ransomware. *Computer Fraud & Security*, 2020(1):14–19, 2020.

[78] Jamie Pont, Osama Abu Oun, Calvin Brierley, Budi Arief, and Julio Hernandez-Castro. A roadmap for improving the impact of anti-ransomware research. In *Nordic Conference on Secure IT Systems*, pages 137–154. Springer, 2019.

[79] Nathaniel Popper. Ransomware attacks grow, crippling cities and businesses. https://www.nytimes.com/2020/02/09/technology/ransomware-attacks.html, Feb 2020.

[80] Subash Poudyal, Kul Prasad Subedi, and Dipankar Dasgupta. A framework for analyzing ransomware using machine learning. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1692–1699, 2018.

[81] Steve Ranger. Locky ransomware: How this malware menace evolved in just 12 months. https://www.zdnet.com/article/locky-ransomware-how-this-malware-menace-evolved-in-just-12-months/, Dec 2016.

[82] Matilda Rhode, Pete Burnap, and Kevin Jones. Early-stage malware prediction using recurrent neural networks. *computers & security*, 77:578–594, 2018.

[83] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In Diego Zamboni, editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[84] Kevin Savage, Peter Coogan, and Hon Lau. The evolution of ransomware. https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware.pdf, Aug 2016.

[85] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 303–312, 2016.

[86] Alex Scroxton. Facilities firm iss world crippled by ransomware attack. https://www.computerweekly.com/news/252478890/Facilities-firm-ISS-World-crippled-by-ransomware-attack, Feb 2020.

[87] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[88] Manish Shukla, Sutapa Mondal, and Sachin Lodha. Poster: Locally virtualized environment for mitigating ransomware threat. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1784–1786, 2016.

[89] Uri Sternfeld. Cerber ransomware variants now actively try to detect and evade canary files. https://www.cybereason.com/blog/cerber-ransomware-vaccine, August 2017.

[90] Kurt Thomas, Danny Huang, David Wang, Elie Bursztein, Chris Grier, Thomas J Holt, Christopher Kruegel, Damon McCoy, Stefan Savage, and Giovanni Vigna. Framing dependencies introduced by underground commoditization. 2015. https://www.inwyrd.com/blog/wp-content/uploads/2010/03/weis2015_blackmarket.pdf.

[91] Trend Micro. After wannacry, uiwix ransomware follow suit. https://blog.trendmicro.com/trendlabs-security-intelligence/wannacry-uiwix-ransomware-monero-mining-malware-follow-suit/, May 2017.

[92] Trend Micro. Bug in ryuk ransomware's decryptor can lead to loss of data in certain files. https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/bug-in-ryuk-ransomware-s-decryptor-can-lead-to-loss-of-data-in-certain-files, Dec 2019.

[93] Aragorn Tseng, Y Chen, Y Kao, and T Lin. Deep learning for ransomware detection. *IEICE Tech. Rep.*, 116(282):87–92, 2016.

[94] University of California San Francisco. Update on it security incident at ucsf. https://www.ucsf.edu/news/2020/06/417911/update-it-security-incident-ucsf, Jul 2020.

[95] R Vinayakumar, KP Soman, KK Senthil Velan, and Shaunak Ganorkar. Evaluating shallow and deep networks for ransomware detection and classification. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 259–265. IEEE, 2017.

[96] VIPRE Labs. Unlocking the lockergoga ransomware and what makes it unique. https://labs.vipre.com/unlocking-the-lockergoga-ransomware-and-what-makes-it-unique/, Aug 2019.

[97] Davey Winder. Beware of this new windows 10 ransomware threat hiding in plain sight. https://www.forbes.com/sites/daveywinder/2020/03/05/beware-of-this-new-windows-10-ransomware-threat-hiding-in-plain-sight/#2444d4cc2958, Mar 2020.

[98] Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani. The rise of ransomware and emerging security challenges in the internet of things. *Computer Networks*, 129:444–458, 2017.

[99] Seunghyun Yoon, Taejin Ha, Sunghwan Kim, and Hyuk Lim. Scalable traffic sampling using centrality measure on software-defined networks. *IEEE Communications Magazine*, 55(7):43–49, 2017.

[100] Ilsun You and Kangbin Yim. Malware obfuscation techniques: A brief survey. In *International conference on broadband, wireless computing, communication and applications*, pages 297–300. IEEE, 2010.

[101] Adam Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 129–140. IEEE, 1996.

[102] Hanqi Zhang, Xi Xiao, Francesco Mercaldo, Shiguang Ni, Fabio Martinelli, and Arun Kumar Sangaiah. Classification of ransomware families with machine learning based on n-gram of opcodes. *Future Generation Computer Systems*, 90:211–221, 2019.