2016 IEEE 36th International Conference on Distributed Computing Systems

CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data

Nolen Scaife
University of Florida
scaife@ufl.edu

Henry Carter
Villanova University
henry.carter@villanova.edu

Patrick Traynor
University of Florida
traynor@cise.ufl.edu

Kevin R.B. Butler
University of Florida
butler@ufl.edu

***Abstract*—Ransomware is a growing threat that encrypts a user's files and holds the decryption key until a ransom is paid by the victim. This type of malware is responsible for tens of millions of dollars in extortion annually. Worse still, developing new variants is trivial, facilitating the evasion of many antivirus and intrusion detection systems. In this work, we present CryptoDrop, an early-warning detection system that alerts a user during suspicious file activity. Using a set of behavior indicators, CryptoDrop can halt a process that appears to be tampering with a large amount of the user's data. Furthermore, by combining a set of indicators common to ransomware, the system can be parameterized for rapid detection with low false positives. Our experimental analysis of CryptoDrop stops ransomware from executing with a median loss of only 10 files (out of nearly 5,100 available files). Our results show that careful analysis of ransomware behavior can produce an effective detection system that significantly mitigates the amount of victim data loss.**

I. Introduction

Encrypting ransomware (a.k.a. crypto ransomware) attempts to extort users by holding their files hostage. Such ransomware differs from other types of malware in that its effects are reversible only via the cryptographic keys held by a remote adversary. Users can only regain access to their files through the use of anonymous payment mechanisms (e.g., Bitcoin), further frustrating efforts to take down these campaigns. While this class of malware has existed for well over a decade, its increasingly widespread use now causes tens of millions of dollars in consumer losses annually [37]. Compounding this problem, an increasing number of law enforcement agencies have also been the victim of ransomware [4], [18], losing valuable case files and forcing these organizations to ignore their own advice and pay the attackers. As such, ransomware represents one of the most visible threats to all users.

Combating ransomware is difficult for a number of reasons. First, this malware is easy to obtain or create [48] and elicits immediate returns, creating lucrative opportunities for attackers. Second, the operations performed by such malware are often difficult to distinguish from those of benign software. Finally, ransomware often intentionally targets unsophisticated users who are unlikely to follow best practices such as regular data backups. Accordingly, a solution to automatically protect such users even in the face of previously unknown samples is critical.

In this paper, we make the following contributions:

- **Develop an early-warning system for ransomware:** CryptoDrop is fundamentally different from existing methods of detecting ransomware, which inspect programs and their activity for malicious characteristics. Our system is the first ransomware detection system that monitors user data for changes that may indicate transformation rather than attempting to identify ransomware by inspecting its execution (e.g., API call monitoring) or contents. This allows CryptoDrop to detect suspicious activity regardless of the delivery mechanism or previous benign activity. Our system does not attempt to prevent all files from loss and is not intended to replace a user's normal anti-malware software; rather, CryptoDrop is designed to be effective *even when the user's anti-malware software has failed to block the malware*. Our system is built on Windows, a platform frequently targeted for ransomware attacks, providing a realistic solution to "in-the-wild" threats. In doing so, we attack the core behavior of ransomware in a novel and practical manner that other anti-malware technologies fundamentally cannot.
- **Identify three primary indicators suited to detect malicious file changes:** These indicators each measure an aspect of a file's transformation, and when all three have manifested, a ransomware file transformation has likely occurred. This union indication assists CryptoDrop in reliably detecting ransomware while incurring few false positives. These indicators have not been previously employed in a ransomware detection system, and our analysis of their effectiveness in isolation and unison provides insight into the ability to detect ransomware.
- **Perform most extensive analysis of encrypting ransomware to date:** Demonstrate a 100% true positive rate over 492 distinct ransomware samples across 14 families after as few as 0 and a median of 10 (0.2%) files lost from our test corpus. Finally, we discuss the observed behavior of our samples and discuss how CryptoDrop remains robust despite the significant behavioral differences between families. Through reduction of the number of files lost, we demonstrate that CryptoDrop reduces the need for the victim to pay the ransom, choking attackers' revenue and rendering the malware ineffective.

The remainder of the paper is structured as follows: In Section II, we perform a literature analysis. In Section III, we define and classify ransomware behaviors, our system's indicators, and demonstrate how these are insufficient for fast detection in isolation. Section IV details CryptoDrop's implementation and its scoring and detection mechanisms. In Section V, we obtain live ransomware samples, demonstrate CryptoDrop's effectiveness against real-world attacks, and analyze the behavior of the samples. We conclude in Section VI.

1063-6927/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDCS.2016.46

II. Related Work

Existing techniques to detect malware attempt to classify a given program as malware and stop it using two properties: what the malware *is* and what the malware *does* [38]. Anomaly detection IDS systems use various machine learning and statistical techniques to determine whether a program is performing atypical operations [17], [8], [23], [49], [29], [43], but are often prone to false positives [5], [21], [47].

Signature matching, commonly found in most modern antivirus and IDS deployments, analyzes programs based on known malware characteristics and flags those that match previously observed intrusions. Early signature detection systems used a variety of features to detect malicious code [28], [45], and over years of development the characteristics in modern malware signatures make this technique for classifying known malware extremely accurate. However, malware that has not been previously observed is difficult to identify in these systems. Furthermore, recent research has shown that evading signature detection is possible with relative ease when the malware signatures used are too rigid [9], [24], [32], [33]. While combining multiple IDS suites using different techniques may provide some added accuracy [36], it is still possible to use automated malware packing techniques to evade tiered anti-malware products [35], [46]. Rather than matching known signatures of programs, file integrity monitors such as Tripwire [26] alert the administrator when system-critical files are modified. These monitors are based on simple hash comparisons and fail to distinguish between legitimate file accesses and malicious modifications. Such integrity checks are primarily effective for files that rarely change; user data is expected to change frequently. Accordingly, this type of integrity monitoring is likely to be noisy and frustrate the user.

Recent work by Kharraz et al. explored several different types of ransomware, including those that hold the operating system hostage or steal information [25]. These other kinds of ransomware are indeed a nuisance, but these types of ransomware can be remedied by wiping the system or removing the disk and extracting the user's important data. However, when the user's data is encrypted (or deleted), these simple mitigations no longer apply and may prevent the victim from paying the ransom and recovering his/her files. Andronio et al. developed an analysis tool for detecting Android ransomware through a combination of static and dynamic analysis techniques including monitoring for encryption calls and threatening ransom messages [3]. This system is well-suited for mobile platforms, where applications can often be analyzed in depth before being posted to an app market [30], [7], [34]. On traditional desktop operating systems, however, these techniques would be easily evaded and introduce unacceptable delays in application installation or launch.

Our work differs from these works because our system monitors the user's data for changes instead of directly inspecting the program making the changes. Through this insight, we build an early-warning system for ransomware that can detect a malicious program through examining how the user's data is changing. In this paper, we refer to encrypting ransomware as "ransomware" and focus on remedying this problem.

Fig. 1: The ransom demand from `TeslaCrypt`. The victim is instructed to access a Tor hidden service and pay the ransom with Bitcoin. Some variants allow the victim to recover a small number of files for free as a gesture of goodwill.

III. Ransomware Indicators

Encrypting ransomware works by obfuscating the contents of user files, often through the use of strong encryption algorithms. Victims have little recourse other than paying the attacker to reverse this process [4], [6], [11]. Some attackers even enforce strict deadlines and host elaborate customer service sites to encourage victims to pay [50]. Figure 1 shows an example of one such extortion demand.

The ease with which ransomware can be written and obfuscated limits the effectiveness of traditional signature-based detection schemes [36]. Signature-based detection does little to stop ransomware variants that take advantage of programmatic input via rogue USB devices [13], which can be attached to a system, and automatically open a terminal, type, and execute a program without writing malicious software to the disk. Such attack vectors fundamentally evade traditional anti-malware and application whitelisting systems by avoiding their inspection point, execution from the disk. Solutions to these attacks [44] require OS modification and are not widely deployed. A more robust solution would be based on the detection of the bulk transformation of a user's data *before it completes*, allowing the user to stop such transformation and denying ransomware access to the totality of the user data. This "data-centric" approach minimizes the pressure to pay an adversary as the data loss can be minimized.

The signature behavior of ransomware is its encryption of the victim's data. Ransomware *must* read the original data, write encrypted data, and remove the original data to complete this transformation. Note that detecting calls to encryption libraries alone is not sufficient as many variants implement their own versions of these algorithms. The specific activities that ransomware performs can be refined into three classes:

Class A ransomware overwrites the contents of the original file by opening the file, reading its contents, writing the encrypted contents *in-place*, then closing the file. It may optionally rename the file. **Class B** ransomware extends Class A, with the addition that the malware moves the file out of the user's documents directory (e.g., into a temporary directory). It then reads the contents, writes the encrypted contents, then

moves the file back to the user's directory. The file name when moving back to the documents directory may be different than the original file name. Since the destination file name may not match the original during any move, the state of the file must be carefully tracked each time a file is moved. **Class C** ransomware reads the original file, then creates a new, independent file containing the encrypted contents and deletes or overwrites (via a move) the original file. This class uses two independent access streams to read and write the data.

Additional operations may be performed to frustrate recoverability. For example, ransomware family `TeslaCrypt` disables and removes the Windows volume shadow copies [14] and other variants wipe the disk's free space. We ignore these operations because they do not directly alter user data.

Capturing the data-centric behavior of ransomware is critical to developing an effective detector. We have identified three primary and two secondary indicators through extensive analysis. Below, we discuss the technical details of each indicator and their relevance to ransomware. We then explore how the *union* of such indicators — that is, we require all primary indicators to be present — creates a strong detector with low false positives.

We selected these indicators because they broadly cover the transformation that encrypting ransomware performs on user data. Other indicators of compromise, such as suspicious strings, network traffic [41], and monitoring for ransom demand windows [15], [3] are outside the scope of CryptoDrop since these indicators may not occur or may become obvious too late to protect the user's data. We instead focus on the task of *protecting the user's data from total loss*, which requires our system to monitor this data as it changes. It may therefore be helpful to think of these indicators as hints that a file may be undergoing a transformation from usable to unusable.

A. File Type Changes

The type of data stored in a file can be approximated using "magic numbers." These signatures describe the order and position of specific byte values unique to a file type, not simply the header information. Since files generally retain their file type and formatting over the course of their existence, bulk modification of such data should be considered suspicious.

The `file` utility is a popular program for determining file type. The default "magic" database library contains hundreds of file type signatures, ranging from specific programs ("Microsoft Word 2007+") to general content ("Unicode text, UTF-7"). With this tool, we can track the file type both before and after a file is written. If this type changes, we can infer that some transformation has occurred. However, we note that a single change in file type does not automatically imply malicious actions. For example, when upgrading to a new software version, the application may update the document's format to comply with a new standard.

B. Similarity Measurement

Strong encryption should produce output that provides no information about the plaintext content. Accordingly, we assume that the output of ransomware-encrypted user data is completely dissimilar to its original content. Such meaningful changes to content can be captured through the use of similarity-preserving hash functions [27], [40]. These hashes differ from traditional hash functions because they contain some information about the source file in their output. Through measuring the similarity of two versions of the same file, we can also gain information about *dissimilarity*.

We selected `sdhash` [40] for this metric. This function outputs a *similarity score* from 0 to 100 that describes the confidence of similarity between two files. `sdhash` assists in determining if two files are homologous with a score of 100 indicating a high likelihood that two files are related. Conversely, the authors of this algorithm note that a confidence score of 0 is "statistically comparable to that of two blobs of random data," and this provides a key insight to how a ransomware-encrypted file should be scored. Given the similarity hash of the previous version of a file, a comparison with the hash of the encrypted version of that file should yield no match, since the ciphertext should be indistinguishable from random data. We should therefore obtain a near-zero score when comparing an original copy of a user's file to a ransomware-encrypted version of that file.

C. Shannon Entropy

Entropy is a simple indicator that provides information about the uncertainty of data. Some types of data, such as encrypted or compressed data, are naturally high entropy. Intuitively, a ransomware attack should result in *consistently high* entropy output as the malware reads the victim's files and writes the encrypted content. The Shannon entropy of an array of bytes can be computed as the sum:

$$e = \sum_{i=0}^{255} P_{B_i} \log_2 \frac{1}{P_{B_i}}$$

for $P_{B_i} = \frac{F_i}{totalbytes}$ and F_i, the number of instances of byte value i in the array. This produces a value from 0 to 8, where 8 represents a perfectly even distribution of byte values in the array. Encrypted files will tend to approach the upper bound 8, since each byte in a ciphertext should have a uniform probability of occurring. Others have previously proposed the use of entropy in the space of malware detection [42], [39], [31], however these works focus on classifying the *malware sample* and not on indicating user data transformation.

D. Secondary Indicators

While the primary indicators provide the strongest suggestion of file modification by ransomware, two additional indicators fill gaps left by the primary indicators:

Deletion is a basic filesystem operation and is not generally suspicious. For example, applications often create and delete temporary files as part of normal operation. However, the deletion of many files from a user's documents may indicate malicious activity. Class C ransomware uses file deletion instead of overwriting an existing file to dispose of the original content. This class of ransomware performs a high number of these operations; early detection of this type of malware depends on capturing this operation.

File type funneling occurs when an application reads an unusually disparate number of files as it writes. Applications that read multiple file types but write only a single type

during an execution are not uncommon. A word processor, for example, may allow the user to embed various file types (e.g., pictures and audio) but will typically only write a single file type (the output document). Ransomware takes this innocuous case to an extreme. As ransomware encrypts and writes data, we expect to see a smaller number of output file types. By tracking the number of file types a process has read and written, the difference of these can be assigned a threshold before considering it suspicious.

E. Union Indication

While exploring the behavior of ransomware, we observed that *none* of the benign programs we tested triggered *all three* of our primary ransomware indicators, while the vast majority of ransomware samples did. We demonstrate in Section V that this union of primary indicators is crucial for early detection. Although each indicator provides value in isolation, we use union indication to take action faster.

F. Indicator Evasion

Malware detection is an arms race. As defenders provide mitigations, adversaries will modify their techniques. We expect the same to occur in the ransomware space. However, evading the union of our three primary indicators will require significant effort on the part of an adversary. For instance, while padding a file with low entropy bits may cause our detector to miss it, such behavior will also concurrently skew similarity hashes. Attempts to make output files appear to maintain their original format may also be possible, but not without dramatically skewing the remaining two indicators. Accordingly, we posit that this work is a contribution because it raises the bar significantly compared to existing ransomware detection techniques, requires that future adversaries have much a deeper understanding of modern file systems and formats, and forces them to make very difficult engineering trade-offs to successfully evade all detectors.

IV. Implementation

CryptoDrop focuses on detecting ransomware through monitoring the *real-time change* of user data. The union of these individual indicators provides a strong measure of suspiciousness of a process. By tracking these indicators and monitoring for this condition in a single running process over time, we can develop a reputation score that indicates whether or not the program is likely behaving maliciously. Once a threshold score is reached, CryptoDrop alerts the user and suspends the suspicious process (or family of processes). In this way, we can prevent ransomware from completely encrypting a victim's files, and contain the amount of damage in the event of infection. The primary challenge in constructing this system is detecting ransomware early while limiting false positives that could make the system impractical to use.

A. Reputation Scoreboard

CryptoDrop monitors read and write accesses to the user's protected directories (e.g., "My Documents"). Whenever a filesystem operation exhibits questionable characteristics that trigger CryptoDrop's indicators, it increments the overall reputation score for that process. Once this score reaches a malicious threshold, our system pauses disk accesses for the flagged process and requests permission from the user to allow the process to continue. This system allows us to observe the cumulative behavior for each process over time, and allows us to watch for combinations of suspicious indicators rather than relying on any one indicator in isolation to catch ransomware.

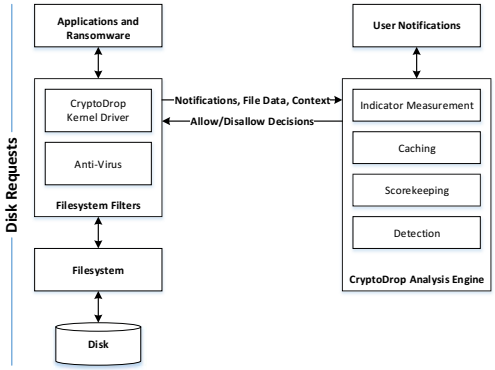


Fig. 2: The architecture of CryptoDrop on Windows. The software filters filesystem reads and writes on user document folders using a filesystem filter similar to other anti-malware solutions. The ordering of the filesystem filter drivers is dependent on the other drivers installed and does not affect our system. Requests are then forwarded to the analysis engine where measurement, caching, and scorekeeping occurs.

B. Union Indication Scoring

The occurrence of all three primary indicators provides strong insight into a process's behavior. In our testing, no legitimate process triggered the union indicator, which we discovered is reliable for immediately detecting ransomware and halting its disk operations.

This scoring mechanism allows us to keep our scoring thresholds low without incurring significant false positives, in a similar manner to the BotHunter IDS by Gu et al. [20], which was able to observe zero false positives in a real deployment. Because anomaly detection research has shown that the probability of a benign process incurring a false positive on a single detection mechanism is high [5], we maintain a reputation score threshold for all processes. However, the probability of a benign process activating a false positive on all three indicators is much lower (specifically, it is the product of the probabilities of a false positive on each individual detector). CryptoDrop's reliability is reinforced by ransomware's typical behavior, which frequently triggers all three primary indicators.

C. State Tracking and Score Assessment

Figure 2 illustrates the CryptoDrop architecture. We instrument the Windows kernel using a driver that allows us to interpose on calls between processes and the filesystem driver. Such an approach allows us to not only detect when files are changed, but also to protect against the modification of our mechanism by malware. This approach also prevents interference with disk encryption systems (e.g., BitLocker), which operate between the filesystem and the disk.

1) Entropy Measurement: Ransomware often writes ransom payment instructions into new text files in every directory. We found that these small, low-entropy writes over-influence basic averaging of the entropy. Through capturing the entropy of the atomic `read`/`write` operations, a weighted arithmetic mean of these entropy measurements is computed, where the weight w is defined $w = 0.125 \times \lfloor e \rceil \times b$, where b is the total number of bytes in the operation and $\lfloor e \rceil$ is the entropy value rounded to the nearest integer. The constant normalizes the weight to a value from 0 to 1. This weighting ensures that low-entropy and small read/write operations do not over-influence the mean and provides a metric that captures a process's behavior over time.

When a protected file is read or written, we calculate the entropy of the bytes involved in this operation and update the respective weighted average P_{read} or P_{write} for the process performing the operation. After each update of a process's averages, if a process has performed at least one read and one write, we calculate the difference of these means as $e_\Delta = P_{write} - P_{read}$, where $e_\Delta \geq 0$. This delta determines the extent that write entropy has exceeded read entropy. When the system exceeds the threshold $e_\Delta \geq 0.1$, we consider the operation suspicious. This measurement is stateless with regard to the previous or future state of a file and occurs for every atomic read or write operation where the threshold is exceeded. While this threshold is small compared to the total range of possible entropy values (0–8), this value provides resolution for detecting the small entropy increase for compressed files (as discussed in Section III).

V. Experiments

We now demonstrate how CryptoDrop detects and stops ransomware with a low number of user files lost and with few false positives. As we will discuss below, union indication provides high-speed detection capabilities for CryptoDrop without increasing the number of false positives.

A. Experimental Setup

Since ransomware attacks user data, we first built a set of data representative of measured user document directories. Using studies that have examined the distribution of file types over an entire filesystem [16], [2] and over user document directories [22], we constructed a document corpus of 5,099 files spread over a nested directory tree with 511 total directories. First, we aggregated the data from the complete set of 10 Govdocs1 Corpus [19] threads[1], the Govdocs1 file set of `.docx`, `.xlsx`, and `.pptx` files, the OPF Format Corpus [1], and Coldwell's audio comparison files [10]. The combination of these corpora contained 11,809 files. We then approximated the proportions of file types needed for our subset corpus and categorized each file such that we could make selections out of the categories, modeling the results in [22]. For each category, we made random selections from the resulting data set and placed them into each directory of our modeled corpus. Finally, the directory tree was placed into the user's documents folder in a Cuckoo Sandbox [12] guest virtual machine running Windows 7 SP1 (64-bit) with 2GB of RAM. Each Windows guest was configured with its firewall, anti-virus, and user account control disabled in order to remove impediments to successful malware execution. All tests were performed as a single administrative user. For safety, network access was restricted on the host's network stack to prevent the spread of malware. Each virtual machine was reverted to a previous snapshot between samples to remove any possibility of experiments influencing each other. Finally, this virtual machine was instrumented with CryptoDrop and configured with a non-union detection threshold of 200.

We obtained 2,663 malware samples from VirusTotal using ransomware-related search terms and known variant names. The fact that these samples were labeled as ransomware by one or more anti-virus engines does not imply that those samples actually are *encrypting ransomware* or that they will perform *any* operations to attack a victim's data (e.g., they may simply lock the screen). Additionally, ransomware often requires online infrastructure, and this requirement is a heavy burden on the future usefulness of any given sample. If command-and-control servers are taken offline, the ransomware cannot complete its task. Some samples may be mislabeled by anti-virus as ransomware, some may operate once and never again, and others may detect our virtual environment. As a result, a significant part of our testing required culling inert, mislabeled, and corrupt samples from our test set.

Since ransomware generally attacks user data quickly after execution begins, we ran each sample for 20 minutes or until detection occurred. We then verified the SHA-256 hashes of the documents to ensure they were present and unmodified. This initial run of all samples took over 22 days to complete. 189 samples that had started to attack the provided documents but had not completed in the time allotted were restarted with a one hour timeout. If no detection occurred and no files were modified, we marked the sample as "inert" and excluded it from future trials. Note that CryptoDrop does not have a detection timeout and is not affected by the run time of malware; the timeout was in place merely to prevent spending excess time on unusable samples. In total, 2,171 malware samples were removed from the set. No removed samples had modified any user files.

CryptoDrop detects ransomware based on its behavior against user data, so the number of families of ransomware we test is *more important* than the raw number of samples. Each sample we test of the same family is unlikely to exhibit significantly different behavior than previous ones, ultimately leading to skewed results. Of the various types of ransomware, Kharraz et al. discussed four unique variants of encrypting ransomware [25]. We collected 14 distinct families of ransomware, including all four previously-discovered families, making ours the most comprehensive study of encrypting ransomware to date.[2] Table I shows the breakdown of the samples by family and class, including the number of files lost before detection. Due to the breadth of families that CryptoDrop detected, we believe that the above methodology successfully culled unusable samples from our original set.

[1]Each of the 10 Govdocs1 "threads" consist of a randomly-selected 1,000 files from the complete Govdocs1 corpus. These were preselected by the Govdocs1 maintainers for use in research projects.

[2]One sample, known to McAfee as `Ransom-FUE`, was tested but excluded from family counts because different anti-malware products mark it as generic malware or belonging to disparate ransomware families.

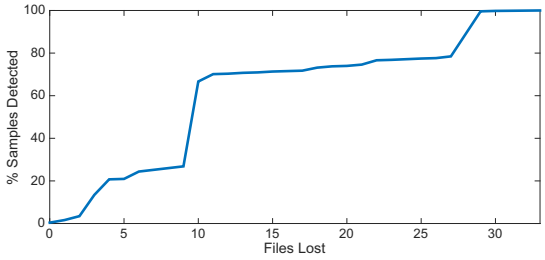


Fig. 3: The cumulative percentage of samples detected at each sample's number of files lost before detection. The median number of files lost before detection was 10, and CryptoDrop detected all 492 samples with 33 or fewer files lost.

We confirm in this data that there is a relative lack of diversity within each family's behavior. In both the `TeslaCrypt` and `CTB-Locker` families, which consist of over half of the working samples we obtained, two or fewer samples showed behaviors beyond their families' primary behavior class. The greatest diversity was found in the `Filecoder` and `CryptoLocker` families, though these two family names are often used as generic ransomware detection names. As antivirus detection names evolve, samples in these two families may be further refined. Due to the homogeneous nature of the behavior in each sample, our data shows that CryptoDrop remains robust against many forms of encrypting ransomware despite low counts of usable samples in some families. *Because our study covered nearly four times the number of families of the previous study [25] and there was little diversity within families, there was little need to collect additional samples.*

B. Ransomware Detection

CryptoDrop detected all the remaining 492 samples, quickly protecting the majority of victim's data with *as few as zero files* encrypted before detection.[3] This result highlights the required actions of ransomware and the effectiveness of our indicators at detecting this type of malware. Below, we discuss the ability of our system to protect user data and the effectiveness of union indication.

1) Data Loss: The amount of data lost before detection occurs is our most valuable metric. When CryptoDrop is able to detect the ransomware more quickly, fewer files are encrypted by the ransomware, thus protecting more of the user's data. Figure 3 shows the data loss as a cumulative plot of the number of files lost before detection. The number of files lost in each case is dependent on the particular variant and the order in which it attacks files. For example, samples which attack high entropy files first (e.g., `.docx`) experience a delay before being assigned points for increasing file entropy. These samples perform high-entropy reads early, resulting in a small delta between read and write entropies, but as these samples move to other files in the user's documents, this advantage quickly disappears. Furthermore, some samples do not prioritize the same file types as others. This behavior has no effect on CryptoDrop's ability to detect malware, as the primary indicators quickly uncover the ransomware.

[3]Two Class C samples created new files but did not successfully remove the original files. For more information, see Section V-C.

Family	# Class A	# Class B	# Class C	Total	Median FL
CryptoDefense			18	18 (3.66%)	6.5
CryptoFortress	2			2 (0.41%)	14
CryptoLocker	13	16	2	31 (6.30%)	10
CryptoLocker (copycat)		1	1	2 (0.41%)	20
CryptoTorLocker2015	1			1 (0.20%)	3
CryptoWall	2		6	8 (1.63%)	10
CTB-Locker	1	120	1	122 (24.80%)	29
Filecoder	51	9	12	72 (14.63%)	10
GPcode	12		1	13 (2.64%)	22
MBL Advisory			1	1 (0.20%)	9
PoshCoder	1			1 (0.20%)	10
Ransom-FUE[2]		1		1 (0.20%)	19
TeslaCrypt	148		1	149 (30.28%)	10
Virlock			20	20 (4.07%)	8
Xorist	51			51 (10.37%)	3
# Samples	**282 (57.32%)**	**147 (29.88%)**	**63 (12.80%)**	**492 (100%)**	**10**

TABLE I: This table shows the breakdown of the 492 ransomware samples that CryptoDrop detected across each class and its median files lost result. We matched each sample CryptoDrop detected with a family name chosen by popularity from detection names from 57 antivirus vendors.

In the median case, the system detected ransomware after only 10 of the 5,099 test files (0.2%) were lost. We observe that Class B samples had the highest number of files lost, and we found that this is due to these samples attacking the user's smallest documents first. We discuss this in detail in Section V-C. By protecting the majority of the user data, our robust approach outperforms traditional anti-malware which allows the complete encryption of the data to occur if the ransomware is not immediately detected.

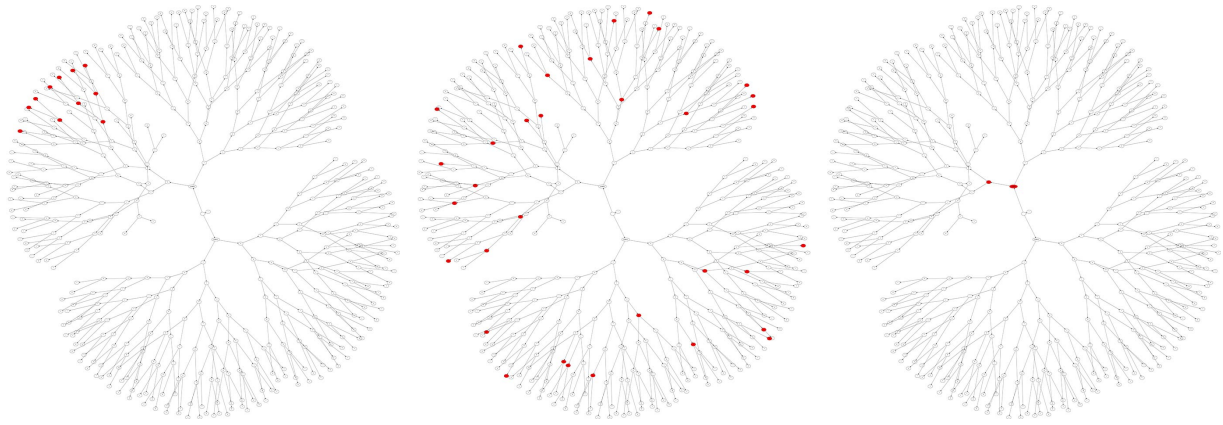
2) Union Indicator Effectiveness: All three primary indicators proved valuable in the majority of samples, with 457 (93%) having at least one occurrence of union indication. We observe that this indicator enables CryptoDrop to detect samples in as few as *one file lost* due to this both dramatically increasing the current score of a process and lowering that process's detection threshold. Accordingly, the ransomware is able to encrypt fewer files before detection.

We found our sample set included 63 samples of Class C ransomware, which can evade union indication by writing its data into separate files. However, 41 of these moved the new file over the original content, allowing linking the original and new content and ultimately leading to union detection. The remaining 22 samples evaded union detection but were unable to evade overall detection due to the large number of high-entropy writes and deletes it performed. CryptoDrop's low non-union threshold mechanism allowed these samples to be detected with a median loss of 6 files.

The 13 remaining samples that did not trigger union indication (but were detected) were all Class A and did not have a similarity indicator alert before detection, preventing union detection on these samples. This does not mean that the malware successfully evaded the similarity indicator, only that CryptoDrop detected the sample before the similarity indicator triggered. The experimental results emphasize CryptoDrop's ability to detect *all* samples in a similar number of files. Union indication, however, is critical to accelerating these detections to as few as one file lost.

C. Data-Centric Ransomware Behavior

We initially expected to find that most ransomware samples perform a depth-first search using ordered lists of directories and upon reaching the deepest files, sequentially encrypt them. After the experiments, however, we found differing results



(a) TeslaCrypt (b) CTB-Locker (c) GPcode

Fig. 4: This tree represents the directory tree we filled with our modeled test corpus documents. The root of the provided documents directory is located in the center of the tree. The filled (red) nodes represent those directories where at least one of our files was read or written before CryptoDrop detected and stopped the malware. Each of these samples uses a different algorithm for attacking the victim's data. From left to right, these samples are Classes A, B, and C, though the classification of the ransomware is unrelated to the order that it encrypts files.

between samples as shown in Figure 4. Each graph in this figure shows the directory tree (rooted in the center of the graph), with shaded nodes representing those directories where the malware read or wrote a file.

`TeslaCrypt` uses a depth-first search, as shown in Figure 4a. It starts accessing files once it has reached the deepest directory. The sample did not begin encrypting files in the first directory it accessed, instead writing the decryption instructions/ransom demand into that directory. It began encrypting our test corpus with a PDF in the second directory it accessed, despite a PDF being available in the first directory.

`CTB-Locker` is shown in Figure 4b. This sample attacks files with certain extensions (`.txt` and `.md`) in ascending order by file size. The set of files the samples encrypted before detection was composed of files throughout the file corpus we provided the malware; *the sample encrypted the next file in size ascending order regardless of whether or not it existed in a different directory*. This behavior is curious because one might assume that a victim's largest files are the ones that contain the most information and therefore have a higher likelihood of being valuable. Since the malware attacked the smallest files in the corpus, 26 of these files lost have sizes $< \ 512B$, causing an above-average number of files to be lost. `sdhash` is unable to generate similarity scores for such small files, causing union detection to be impossible until the malware has moved past this threshold. To test this hypothesis, we reran one of these samples with a corpus missing all of the files with sizes $< \ 512B$. The number of files lost on this run was 7, far fewer than the 29 previously lost. Through our experimental observations, we believe situations like this can be remedied in future versions of CryptoDrop through automatic identification of conditions unfavorable to individual indicators. Once identified, CryptoDrop could adjust the number of reputation points assessed up or down for individual indicators, leading to faster detection even when union indication is not possible.

`GPcode`, shown in Figure 4c, accesses files starting at the root directory and moving down the tree. This sample is particularly notable because it did not modify or delete any of our test files before being detected. This Class C sample read files from each directory it accessed and wrote a new file. The sample did attempt to delete the original content, but some of our test files were marked read-only on the filesystem, which this sample was uniquely unable to work around. Regardless, CryptoDrop detected this sample due to the high entropy delta between the files it was reading and writing. If this sample had been able to successfully delete the files, the number of files lost would have been ≤ 6. It is surprising that this sample remains functional, as it was first submitted to VirusTotal in 2008. This may also provide some clue as to why this sample had errors with basic file functionality.

Ultimately, the ordering of the files attacked by the malware influences CryptoDrop's speed to detect and stop the ransomware, though we note that there may be future optimizations to the indicators to assist with detecting attacks on corpora of small files. One such optimization might increase the points assessed on certain indicators when others are unavailable. We leave dynamic scoring to future work but note that this may have an adverse effect on false positives.

D. File Format Attack Frequency

We constructed the set of file extensions that each sample attacked and then calculated the frequency of those extensions among all of the samples. For example, if a sample accessed more than one PDF file, the data shows a single PDF access for that sample. This data is shown in Figure 5 and provides a glimpse into ransomware's priority of file formats.

The data indicates a strong preference for attacking productivity files over other kinds of media including pictures and music. It is unclear whether these file formats appear more frequently in the early accesses of ransomware simply because they are common or if they represent data that is more valuable to the victim (and therefore more likely to cause a ransom payment). We leave this discussion for future work, but

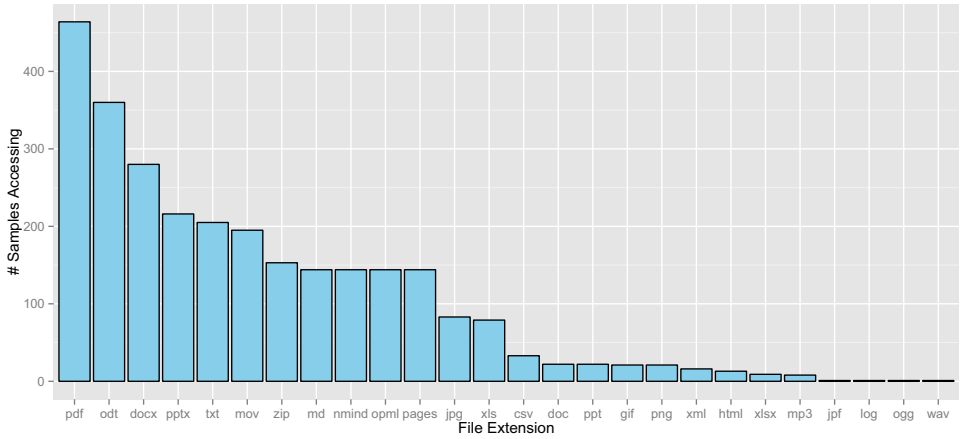


Fig. 5: This figure shows the frequencies of aggregate file extensions accessed by the 492 live ransomware samples we obtained. The data was collected until CryptoDrop detected the sample, causing the data to represent the *first* files attacked by each sample. Overall, the samples attacked common productivity formats first.

note that the top four file formats (`.pdf`, `.odt`, `.docx`, and `.pptx`) represent compressed, high-entropy files. Although these files exhibit far less entropy increase when encrypted than other files (see Section III for details), CryptoDrop's sensitivity to this increase nevertheless enables fast detection.

E. Ransomware Scripts

One family of ransomware we obtained, `PoshCoder`, was developed in PowerShell. This sample was particularly notable because it underlines that ransomware does not need to be a compiled binary. As a result, it can be quickly morphed into an unknown variant and typed or piped directly into an interpreter. Signature-based anti-malware technologies cannot defend against this malware because it is not necessary for this malware to exist on the disk of the victim host. It can be constructed, executed, and completed entirely in memory.

This particular sample continues to have an extremely low detection rate among anti-virus vendors, with only 8 of 57 products detecting the sample as of submission.[4] CryptoDrop, however, detected this sample after only 11 files were lost. To more fully demonstrate the issue of signature-based ransomware detection, we added a single character to the above sample and submitted the file back to VirusTotal. Two of the 6 products which detected the original sample no longer detected the malware. Since CryptoDrop is focused on the changes to user data, not the malware's contents or its execution, our system is well- positioned to stop ransomware which manipulates the filesystem using high-level APIs.

F. False Positives

While CryptoDrop is effective at quickly detecting ransomware, we note that any evaluation of its real-world utility must also include a discussion on incorrect detection of benign activity. False positive analysis for a system such as CryptoDrop is challenging since its analysis requires changes to be made to a user's protected documents. Techniques used in static malware analysis works (e.g., providing a set of known-good binaries alongside bad ones) will not work since CryptoDrop does not analyze binaries for malicious traits. Likewise, techniques used in dynamic malware analysis (e.g., passively observing benign activity on a system and running the detector on it later) will not work since CryptoDrop needs to measure the user's documents before and after each change. Below, we discuss the results of our experiments with benign programs and show that our system remains robust.

We evaluated thirty common Windows applications on the same virtual machine configuration used to test malware samples and found only one false positive. That false positive, 7-zip, was expected as it reads a large number of disparate files and generates high entropy output (similar to ransomware). We discuss mitigating this specific type of false positive in the following section. Furthermore, no application exhibited *all three* primary indicators. Our Windows application data set includes: 7-zip, Adobe Lightroom, Avast Anti-Virus, Chocolate Doom, Chrome, Dropbox, F.lux, GIMP, ImageMagick, iTunes, Launchy, LibreOffice Calc, LibreOffice Writer, Microsoft Excel, Microsoft Office Viewers, Microsoft Word, MusicBee, Paint.NET, PhraseExpress, Picasa, Pidgin, Piriform CCleaner, Private Internet Access VPN, ResophNotes, Skype, Spotify, Sticky Notes, SumatraPDF, uTorrent, and VLC Media Player.

For space limitations, we are unable to thoroughly discuss the testing methods used for all thirty applications. However, to get better insight into how CryptoDrop analyzed benign applications, we analyze five important applications below. Figure 6 shows the number of false positives that would have occurred at varying detection thresholds with these five applications. In our analysis, we demonstrate that our threshold selection minimizes false positives while maintaining fast detection of ransomware. The score of each process at the end of each experiment is listed next to the application name.

Adobe Lightroom (107): We imported a set of 1,073 JPEG image files. We then performed an "automatic tone" function on every picture, converted 5 photos to black-and-white, and exported these 5 photos to the user's documents

[4]We note that this metric *does not* mean that a particular anti-virus product would not detect this sample in real- world conditions. When provided this sample with no additional context, it did not detect.

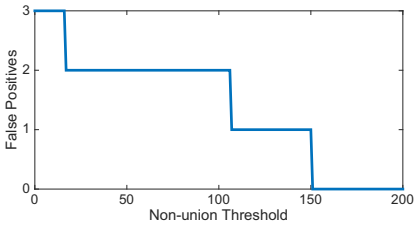


Fig. 6: The number of false positives registered against five top Windows applications for a range of non-union thresholds. During our malware-testing experiments, the threshold for non-union detection was configured at 200.

folder. **ImageMagick (0):** We performed a batch modification of the same 1,073 JPEG image files, using the ImageMagick `mogrify` utility. Each picture was rotated 90 degrees and saved in-place. **iTunes (16):** Before opening iTunes, we first deleted the iTunes library to force it to generate a new one. We then opened iTunes, imported all 70 of the Coldwell audio comparison files [10], and allowed iTunes to convert any files that were unsupported. We played three songs, then converted all of the audio files to AAC format using iTunes built-in conversion function. **Microsoft Word (0):** We created a new blank document and entered 5 paragraphs of text, then saved the file. We then created a table, added one paragraph of text to each cell, adjusted the formatting, and saved the file again. We imported a photo into the file, and saved the file once again. Finally, we inserted a "SmartArt" graphic, added text to it, and saved. **Microsoft Excel (150):** We created a blank document and filled in two 500-cell columns with values. We then created a line chart of these two columns, saved the document, and closed Excel. We re-opened Excel, added another column of values, added a scatter plot of these, and saved the file again.

These scores represent the window of time from process start to the time we completed the task. Though some of the scores may seem higher than expected, we note that this time window is *not* a similar scale to that of the ransomware samples. Due to the extremely aggressive nature of ransomware, we were generally able to detect real ransomware samples seconds after they began accessing user data. Our false positive tests, however, took tens of minutes of high disk activity to complete. Our Lightroom test took *nearly an hour* as the files were copied, indexed, and added to the software's library. Monitoring any time window presents an evasion opportunity to ransomware as it can change its rate of attack to overcome the window. However, research into time window parameterization may lead to another primary indicator in future versions of CryptoDrop.

G. Limitations

CryptoDrop is unable to determine the intent of the changes it inspects. For example, it cannot distinguish whether the user or ransomware is encrypting a set of documents. This fundamental limitation requires our system to notify the user when suspicious activity occurs and allow the user to make the final decision on whether the activity is desired. As a result, we expect that programs such as GPG and PGP, compression applications, and other applications which perform similar transformations will cause a CryptoDrop detection when applied to many user documents. While testing 7-zip, a detection occurred while attempting to create an archive of the user documents directory. We believe that a CryptoDrop detection of this behavior is normal, expected, desirable, and not overwhelmingly invasive. Future versions of ransomware may simply employ the user's own (or a bundled) GPG or compression utility to perform encryption. Since CryptoDrop is looking for bulk transformation, which these applications perform, our system can notify the user of this suspicious activity in case it is malicious.

H. Performance

The initial research version of CryptoDrop is unoptimized. It contains many synchronous calls, verbose logging, debugging measurements, and other performance-decreasing functionality. Compiler optimizations were disabled to ensure correct operation and assist in debugging during our experiments. We traced our code while performing modifications to protected files and observed that this version of CryptoDrop introduced overhead latencies for file open and read operations of less than 1ms. Close operations added an average latency of 1.58ms. Write and rename operations are the most expensive with additional latency of 9ms and 16ms, respectively. CryptoDrop does not inspect files outside of the user's documents directory, so operating system and program accesses to other files (including their own) are not affected.

As an example, the high latency from write and rename operations often manifests during measurement. During these operations, the file is often locked and cannot be opened by CryptoDrop for inspection; when this happens, CryptoDrop switches context and reads the file using the kernel code. Currently, our system writes this data back to temporary files on the disk so it can be manually inspected later for correctness. One major optimization would be to perform these measurements without additional disk access, heavily reducing these latencies. We believe that with future optimizations, CryptoDrop can be run on a live system with a small overhead.

VI. Conclusion

Ransomware continues to plague unsuspecting victims due to its use of strong cryptography. Victims often have little recourse other than to pay the ransom, fueling a vibrant economy for attackers who can deploy new variants with ease. In this paper, we limit attackers and reduce the incentive for victims to pay with CryptoDrop, an early-warning system for ransomware attacks. Our solution targets ransomware by monitoring the victim's data and detecting the behaviors that ransomware must perform. We first identify these required operations, classify ransomware into three major classes, and develop indicators that inspect, capture, and alert on ransomware while avoiding benign applications. We discover that ransomware frequently trips all of these primary indicators, while legitimate applications do not, creating a shortcut to detecting ransomware with fewer files lost. Our experiments test CryptoDrop against 492 real-world ransomware samples (representing 14 distinct families, the largest study of encrypting ransomware to date) and find a 100% detection rate with as few as zero victim files lost before detection. With few files lost, the burden to pay for victims of ransomware is reduced or removed, protecting users and dismantling the economy of attackers.

ACKNOWLEDGMENTS

The authors would like to thank EldoS for access to their driver framework and VirusTotal for providing malware samples. This work was supported in part by the US National Science Foundation under grant numbers CNS-1464088 and CNS-1540217. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] OPF format corpus. http://openpreservation.org/technology/corpora/opf-format-corpus/.

[2] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *ACM Transactions on Storage (TOS)*, 2007.

[3] N. Andronio, S. Zanero, and F. Maggi. HelDroid: Dissecting and detecting mobile ransomware. In *Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID)*, 2015.

[4] E. Arnold. Tennessee sheriff pays ransom to cybercriminals, in bitcoin. http://www.bizjournals.com/memphis/blog/2014/11/tennessee-sheriff-pays-ransom-to-cybercriminals-in.html, 2014.

[5] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 1999.

[6] D. Carrigan. Police departments hit by ransomware virus. http://www.wcsh6.com/story/news/local/2015/04/10/police-departments-hit-by-ransomware-virus/25593777/, 2015.

[7] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck. MAST: Triage for market-scale mobile malware analysis. In *Proceedings of ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. ACM, 2013.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009.

[9] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *IEEE International Conference on Dependable Systems and Networks*, 2008.

[10] N. Coldwell. Comparison of audio compression. http://nigelcoldwell.co.uk/audio/.

[11] D. Common. Ransomware victims pay cybercriminals to save family photos. http://www.cbc.ca/news/technology/ransomware-victims-pay-cybercriminals-to-save-family-photos-1.2962106, 2015.

[12] Cuckoo Foundation. Automated malware analysis - cuckoo sandbox. http://www.cuckoosandbox.org/.

[13] C. Cuevas and C. Shaffer. Raising the white flag. http://www.shmoocon.org/2012/videos/ShafferCuevas-RaisingTheWhiteFlag.m4v, 2012. Presented at ShmooCon.

[14] T. Dewan. Teslacrypt joins ransomware field. https://blogs.mcafee.com/mcafee-labs/teslacrypt-joins-ransomware-field, 2015.

[15] C. J. Dietrich, C. Rossow, and N. Pohlmann. Exploiting visual appearance to cluster and detect rogue software. In *Proceedings of the ACM Symposium on Applied Computing*. ACM, 2013.

[16] J. R. Douceur and W. J. Bolosky. A large-scale study of file-system contents. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 1999.

[17] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 1996.

[18] B. Fraga. Swansea police pay $750 "ransom" after computer virus strikes. *The Herald News*, 2013.

[19] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6, Supplement(0), 2009.

[20] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: detecting malware infection through IDS-driven dialog correlation. In *Proceedings of the USENIX Security Symposium*, 2007.

[21] D. Hadžiosmanović, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle. N-gram against the machine: On the feasibility of the N-gram network analysis for binary protocols. In *Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID)*, 2012.

[22] B. J. Hicks, A. Dong, R. Palmer, and H. C. Mcalpine. Organizing and managing personal electronic files: A mechanical engineer's perspective. *ACM Transactions on Information Systems (TOIS)*, 26(4), 2008.

[23] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *International Journal of Information and Computer Security*, 6(3), 1998.

[24] S. Jana and V. Shmatikov. Abusing file processing in malware detectors for fun and profit. In *IEEE Symposium on Security and Privacy (S&P)*, 2012.

[25] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. *DIMVA*, 2015.

[26] G. H. Kim and E. H. Spafford. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the ACM Conference on Computer and Communications Security*, 1994.

[27] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3, Supplement(0), 2006.

[28] S. Kumar and E. H. Spafford. A generic virus scanner for c++. In *Proceedings of the Computer Security Applications Conference*, 1992.

[29] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda. AccessMiner: Using system-centric models for malware protection. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2010.

[30] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2013.

[31] R. Lyda and J. Hamrock. Using entropy analysis to find encrypted and packed malware. *IEEE Security and Privacy*, 5(2), 2007.

[32] D. Maiorca, I. Corona, and G. Giacinto. Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, 2013.

[33] J. A. P. Marpaung, M. Sain, and H.-J. Lee. Survey on malware evasion techniques: State of the art and challenges. In *International Conference on Advanced Communication Technology (ICACT)*, 2012.

[34] Y. Nadji, J. Giffin, and P. Traynor. Automated remote repair for mobile malware. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, 2011.

[35] J. Oberheide, M. Bailey, and F. Jahanian. PolyPack: An automated online packing service for optimal antivirus evasion. In *Proceedings of the USENIX Conference on Offensive Technologies*, 2009.

[36] J. Oberheide, E. Cooke, and F. Jahanian. CloudAV: N-Version antivirus in the network cloud. In *USENIX Security Symposium*, 2008.

[37] G. O'Gorman and G. McDonald. Ransomware: A growing menace. Technical report, Symantec Corporation, 2012.

[38] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12), 2007.

[39] R. Perdisci, A. Lanzi, and W. Lee. Classification of packed executables for accurate computer virus detection. *Pattern recognition letters*, 29(14), 2008.

[40] V. Roussev. Data fingerprinting with similarity digests. In *Advances in Digital Forensics VI*, IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2010.

[41] N. Scaife, H. Carter, and P. Traynor. OnionDNS: A seizure-resistant top-level domain. In *In IEEE Conference on Communications and Network Security (CNS)*, 2015.

[42] I. Sorokin. Comparing files using structural entropy. *Journal in Computer Virology*, 7(4), 2011.

[43] A. Tang, S. Sethumadhavan, and S. Stolfo. Unsupervised Anomaly-based Malware Detection using Hardware Features. In *Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID)*, 2014.

[44] D. J. Tian, A. Bates, and K. Butler. Defending against malicious USB firmware with GoodUSB. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, 2015.

[45] P. Traynor, M. Chien, S. Weaver, B. Hicks, and P. McDaniel. Noninvasive methods for host certification. *ACM Transactions on Information and System Security*, 11(3), 2008.

[46] X. Ugarte-Pedrero, D. Balzarotti, I. Santos, P. G. Bringas, and S. Antipolis. SoK : Deep Packer Inspection : A Longitudinal Study of the Complexity of Run-Time Packers. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[47] A. Viswanathan, K. Tan, and C. Neuman. Deconstructing the assessment of anomaly-based intrusion detectors. In *Proceedings of the International Symposium on Research in Attacks, Intrusion and Detection (RAID)*, 2013.

[48] J. Walter. Meet tox: Ransomware for the rest of us. https://blogs.mcafee.com/mcafee-labs/meet-tox-ransomware-for-the-rest-of-us/, 2015.

[49] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 1999.

[50] H. Weisbaum. CryptoLocker crooks launch 'customer service' site. http://www.cnbc.com/id/101195861, 2013.