



On Tracking Ransomware on the File System

Luigi Catuogno¹ ^a and Clemente Galdi²  ^b

¹*Dipartimento di Informatica, Università degli Studi di Salerno, Fisciano, Salerno, Italy*

²*Dipartimento di Studi Politici e Sociali, Università degli Studi di Salerno, Fisciano, Salerno, Italy*

Keywords: Ransomware, Ransomware Detection, Ransomware Tracking, Malice Indicators, File System Hooking, Testbed.

Abstract: Ransomware detection is gaining growing importance in the scientific literature because of widespread and economic impact of this type of malware. A successful ransomware detection system must identify a malicious behaviour as soon as possible while reducing false positive detection. To this end, different strategies have been explored. Recently, a promising approach has risen. It consists in looking for possible running ransomware by measuring the different activities every process does on the filesystem. Such measurements are represented with quantitative “indicators”. Indicators selection and their interpretation, is a critical and challenging task. In this paper we survey some of most representative file-system centered ransomware detectors and describe their chosen behavioural indicators and strategies used to measure them. Then we compare the different solutions and discuss pros, cons and open issues of every approach.

1 INTRODUCTION

Young and Yung (1996) first envisioned in 1996 the threat of crypto-viruses, malware intended to block (using cryptographic algorithms) or steal user data for extortion intents. Since their seminal paper, “ransomware” have turn out as a major plague for desktop computer.

Ransomware are a specific type of malware and require specific strategies to be contained. Indeed, nowadays ransomware increasingly threaten traditional malware prevention systems such as signature-based detection systems and anomaly detection systems (ADS).

On one hand, the number of ransomware that leverage polymorphic code or that introduces some randomization in the code execution (Ma et al., 2012, Milosevic et al., 2016, Oberheide et al., 2009), in order to circumvent signature scanning, is constantly growing. Moreover, new virus variants evolve so rapidly that sometimes the deployment of signature updates gets left behind.

On the other hand, ransomware mimic ordinary processes. Indeed, very often, ransomware samples feature an execution profile quite similar to the one of legitimate processes, so that, it is increasingly dif-


ficult, for ADSEs to identify deviating activities. In this case, raising ADSEs sensitivity may result in an unacceptable growth of false positives (Le, 2015, Viswanathan et al., 2013).


Furthermore, generic malware present a wide range of tasks and objectives that might require a certain time to be achieved while ransomware is immediately dangerous. When a sample is discovered, it is probably too late.

Ransomware pursue a sole plan: encrypting the user data stored on the victim’s computer in order to request the payment of a fee (ransom). To this end, ransomware always perform these actions: run an algorithm to select the target files; open such files and encrypt their content; replace/overwrite their original contents with the corresponding cyphertexts.

Ransomware are designed to encrypt as much data as possible in the shortest possible time (*i.e.*, likely before being detected). Unfortunately, “traditional” Anomaly Detection Systems need to gather some information for a while before making a decision about any suspect ransomware process. In the meantime, the files that have been accessed by the malicious process are gone.

A different approach to ransomware detection can be set up on the basis of some observation. First, ransomware processes always perform the aforementioned operations following essentially the same pat-

^a  <https://orcid.org/0000-0002-6315-4221>

^b  <https://orcid.org/0000-0002-2988-700X>

tern, whereas benign processes may not. Second, unlike benign processes, ransomware usually perform massive file system activity.

Taking advantage by these observations, Kharraz et al. (2016, 2015) proposed to seek evidence of ransomware activities by tracking the victim’s file system operation. So that, whenever any suspect activity is discovered, the detection system identifies the responsible process(es) and takes the appropriate decisions.

Notice that the difference of this new approach with respect to traditional ADSes is twofold: (1) the system looks for those processes whose activity converges to a limited set of expected “malicious behaviors” instead of analyzing any deviation from the presumptive “normal” behavioral model; (2) the detection process focuses on the effects of the malicious code rather than its execution.

Moreover, placing the detector at the file system level has a valuable advantage: detectors are enabled to trace every suspect activity. So that, whenever any malicious action is discovered, it is possible to revert it, as long as a certain data retention capability has been given to the file system. This allows to recover every damage to data the malware have done.

In the last three years, this strategy has witnessed a remarkable development. Several solutions are currently on the stage. Main differences amongst such proposals mainly concern of: (a) the technique each system uses to track per-process file system activity; (b) the characteristics of this activity each system takes in account; (c) the way in which collected information are aggregated and represented as “malice indicators”; how suspect processes are classified according to their indicators and how each system eventually takes a decision.

In this paper, we provide a survey of some of the most representative proposals of “file system centric” ransomware detection systems. We compare the different solutions according to the way each one deals with the aforementioned aspects.

Rather than a mere survey of ransomware detection systems, the main goal of our work is a qualitative analysis of the “toolbox” that has been made available over time to counter the threat of ransomware. We analyze the way each tool is used and how it may affect the overall system performance. Furthermore, we aim at briefly discussing pros and cons of the different solutions along with possible further developments and challenges.

The paper is organized as follows. In Section 2 we introduce the main techniques and frameworks ransomware detectors use to collect information about how suspect processes access the file system. In Sec-

tion 3 we examine the different criteria to select the information to be gathered for each process along with the techniques for *malice indicators* computation and interpretation. Section 4 provides a brief survey of the ransomware detection systems we considered in our study. Section 5 introduces some of main datasets related to ransomware samples activity that have been made available to test and train new detection systems. Section 6 concludes the work.

2 TRACKING RANSOMWARE ON THE FILE SYSTEM

In this section we survey the main techniques and tools that are currently used to gather information about the file system activities of running processes. Hence, we propose a discussion about the way they are employed for ransomware detection and some of the most challenging open issues.

File System API Hooking. Modern operating systems provide development tools and APIs which allow to extend the file system services by adding new features, as well as implementing diagnostic and monitoring functionalities. In particular, in such OSes the *Virtual File System* layer (VFS) features some *hooks* in every system call implementation. This makes possible, for example, to intercept every system call invoked by any user-level process having r/w access to each parameter (including read or to-be-written data buffer), as well as intercepting its results before the call returns. Ransomware detection may leverage such a feature to track whatever a process does on the file system and in the files it opens.

The Windows™ Operating Systems features different levels of file system hooking. In particular, the *File System Minifilter Driver* (Microsoft Inc., 2014) provides a high-level API which addresses monitoring applications, including profiling and antivirus tools.

With Android’s FileObserver (Google Inc., 201x) applications register their *listener* to the file system manager in order to be notified whenever any event such as OPEN, ATTRIB, CLOSE occurs on any file of interest.

Virtual Machine Introspection (VMI). Virtual Machine Introspection (VMI) is a technique firstly proposed by Garfinkel and Rosenblum (2003). In VMI, the *monitored system* (target) runs in a Virtual Machine (VM), whereas a *monitoring application* (monitor) is executed in a privileged VM or

plugged into the hypervisor itself. The monitor has access to every aspect of the target operation through an interface oriented to hardware-level details such as target’s memory page allocation, CPU register status, interrupts, memory access and so on. Nowadays, LibVMI (Payne et al., 2015) is probably one of most successful VMI framework and is widely used for malware analysis.

However, the inspection of the target operating system at a finer-grained level (*e.g.*, tracing *per-process* file system activity) through such low-level interfaces is quite hard due to the so called “*semantic gap*” as argued by Chen and Noble (2001) and More and Tapaswi (2014). To cope with this problem, the monitor should be provided of adequate information concerning the VM/Guest OS internal organization and current status (*e.g.*, the Guest’s kernel symbols map, etc.)

The DRAKVUF toolkit (Lengyel et al., 2014), is built upon LibVMI and features higher level API which help to develop *ad hoc* plug-ins tailored to monitor specific Guest OS-level data structures and events.

Sandboxing. Sandboxing is a technique which makes possible to safely run an untrusted/untested software within a confined execution environment (the sandbox) which resembles the legacy execution environment. Nevertheless, computational resources assigned to the “sandboxed” application are strictly controlled and configurable according the needs of testing.

Running a malicious application into a sandbox, allows to gather useful information about its behaviour ensuring that while executing, it can neither harm the host machine nor spread through the network. It has been shown in Catuogno and Galdi (2016) that different security models, coming from academia and industry essentially identify the same security perimeter, thus guaranteeing the above property being granted independently of the specific security model under evaluation. In case of malware execution. To this end, plenty of sandboxing systems have been proposed. Amongst the most representatives we mention the Cuckoo Sandbox (Claudio Guarnieri, 2011).

We highlight some differences between sandboxes and VMI-based tools. First, sandboxes mainly collect information about “how a given malware sample behaves” instead of “how an infected VM behaves”. Second, sandbox components observe the target process through the OS data structure, whereas VMI collects information at a lower level. Finally, sandboxes feature some components that run in the *same* envi-

ronment of the malware sample in the same environment, while VMI tools are designed to remain stealth (and external) with respect to the guest environment.

2.1 Discussion

Ransomware detection systems (monitors) are used in two different scenarios, each raising different issues and requirements. We denote these scenarios respectively: *operational* and *testbed*. Notice that every detection system can be indifferently used in both scenarios.

In the operational scenario, monitors are used on computers which are actually deployed in their operational environment (*e.g.*, the corporate network). In this case, primary goals are: early detection of any in-progress threat and mitigating its effects/impact.

Performance is a crucial issue. Indeed, the monitor is required not to introduce significant overhead to the system performance, both in terms of throughput and in computational resource consumption.

Table 1 summarizes file system performance overhead claimed by some of most recent proposal addressing the operational scenario and leveraging both virtualization and hooking (such proposals are introduced in Section 4). These values depend on multiple factors and reflect different ways of measuring performance.

At first sight, in presence of virtualization-based monitors, users/applications experience the highest overhead. However such an overhead is mainly due to the virtualization infrastructure rather than the detection services. For example, Gutierrez et al. (2018) highlight that overhead due the file system operation analysis is only a slight percentage (between 0.13% and 1.54%) of the total (which is up to 20%). Moreover, as suggested by Subedi et al. (2017), monitors performance is significantly influenced by the underlying hypervisor.

Hooking-based monitors promise better overall performance as, in principle, throughput slowdown mainly affects file system bound processes. However, beyond the callbacks computational costs, overall performance degradation largely depends on the number of hooked calls and their invocation frequency. Systems like CryptoLock (Scaife et al., 2016) and ShieldFS (Continella et al., 2016) feature the highest overhead as they intercept and “inspect” numerous file system calls. Furthermore, preventing data losses by featuring per-file shadowing and snapshotting mechanisms (such as in ShieldFS and R2D2 (Gutierrez et al., 2018)) increases the latency in those file system operations which entail the creation and removal of different file versions/copies.

Table 1: Ransomware detectors fitting the *operational* application scenario: performance summary. Legenda: waylaying techniques (WT): hooking (H), Virtual Machine (VM).

System	WT	Benchmark	Performance overhead
CryptoLock	H	<i>n/a</i>	open,read: 1ms; close: 1.58ms; write, rename: 9 – 16ms
ShieldFS	H	<i>n/a</i>	“user-perceived” overhead can reach 45 – 75%
DAD	H	WPT, CrystalDiskMark	write, info: 0.011ms, 46 – 82%
R2D2	VM	PCMark 8	1.4 – 9.29% of total latency
RDS3	VM	fio	serial read: 41 – 58%; serial write: 30 – 65% (depending on the VMM)
Redemption	H	IOZone	read: 2.8%; write: 3.4%; rewrite and create: up to 9%
RansomCare	H	<i>n/a</i>	8.7% overhead measured during the execution of a sample task.

Measuring how much these factors affect the file system performance, depends on: (a) file system characteristics including its morphology as well as the amount, the size and the types of files it stores; (b) statistical considerations concerning how monitored processes access stored files.

Furthermore, we point out that it is quite hard doing any quantitative comparison amongst existing anti-ransomware systems due to the variety of benchmarking tools used by the authors (see Table 1). In fact, different benchmarks behave rather differently each other as they stress different aspects of the file systems operation. For example, we have benchmarks which perform bulk r/w operations (*e.g.*, IOZone (Don Capps et al., 2002)) as well as tools which aim at simulating more realistic file system operation such as PCMark (Underwriters Labs LLC, 2013). Nevertheless, we notice that even launching the same test on differently populated file systems may lead to rather different results.

To this regard, we argue that common guidelines in evaluating the performance of anti-malware systems are still due.

In the testbed scenario, the emphasis is on analyzing the behavior of potentially infected systems and suspected samples, in order to acquire knowledge suitable to improve the detection process in the operational scenario. No matter about neither the victim system performance nor the integrity of the fictitious data it stores. Here, the main concern is making the testbed as much *realistic* as possible.

Ransomware attack strategies are significantly driven by the victim’s file system layout. For example, running a sample on an almost empty (or flat) file system might not give useful information about its behaviour if launched on a computer whose hard disk is populated with plenty of working document files and folders. To this end, Kharraz and Kirda (2017) and Continella et al. (2016) modeled their testbed on the basis of the activity of a number of volunteers who worked on fully operational desktop computers for several weeks. However, Palisse et al. (2017) and Scaife et al. (2016) have based their model on several studies concerning file systems population statis-

tics (Agrawal et al., 2007) and sample files collections (Garfinkel et al., 2009).

Transparency (stealthiness) of detection systems has capital importance in both scenarios. Indeed, ransomware are increasing their capability of realizing the presence of detection systems. In such a case, malicious processes may adopt two strategies. On one hand, it attempts to attack the detector, by subverting its code or the system features it relies upon. On the other hand, it may change its behavior (evasion) in order to avoid to be analyzed (Bordoni et al., 2017, Bulazel and Yener, 2017).

3 MALICE INDICATORS FOR RANSOMWARE DETECTION

Once an observation point has been taken, detector engines gather tracks of any operation is done on the file system by every single process in order to recognize those patterns which may lead to malicious activities.

In literature, statistical measurements of the occurrence of such operation patterns are defined as *malice indicators*. Each indicator provides a quantitative representation (score) of the phenomenon it is related to. Indicators roughly fall in two categories (Kharraz and Kirda, 2017): *content-based* and *behavior-based*. The formers (*e.g.*, entropy measurements, similarity, file content r/w and deletion statistics) are related to the way in which the content of each file changes as a result of accessing processes operation. The latter are about measuring the effects that any suspect process activity produces on the whole file system.

This section briefly introduces and discusses some of main proposals about such indicators. Table 2 summarizes the malice indicators considered in some of most recently proposed ransomware detectors.

Read/write Access Statistics. These indicators measure “how” a process reads and/or writes within the files it opens. For example, highly frequent over-

writes of the whole file as well as high frequency of writes performed to multiple files might be an evidence of ransomware activity.

Divergence Measures in Overwritten Buffers. Significant entropy variations within file read/write buffers might entail that a process is overwriting clear/structured data with encrypted data (Lin, 1991). Amongst further measures commonly considered for this purpose, we mention the Kullback-Liebler divergence (Kullback and Leibler, 1951) and the χ^2 goodness-of-fit test (Cochran, 1952, Pont et al., 2020). Performance achieved by leveraging such metrics are rather variable. Content-based indicators are sensible to the type of analyzed files and their configured thresholds. We argue that a comprehensive comparison of the performances achieved by using these indicators is worth of on-the-field

Similarity. Evaluating the *similarity* between r/w buffers may help to discover fraudulent encryption activity. Similarity-aware hash functions (Roussev, 2010) have been proposed to be used in conjunction with the entropy measurement with the aim of reducing false positives.

Moving/Removing Files. Frequency of files moves and removals a process performs on the file system is a behavior potentially related to ransomware activity. In fact, frequently ransomware do not directly overwrite their target file. Instead, this kind of adversary, creates a new file in which it stores the encrypted version of its victim and, eventually, it deletes the original file and replaces it with the encrypted one.

Files Type Modification Rate. The suitability of this indicator relies on the following fact. Data encryption obviously entails that files structured according any type (images, videos, documents, etc.) are changed into files of unknown (or non-existing) type. This is a straightforward consequence of ransomware activity, whereas benign process operation with the same effect is rather unfrequent.

File Types Access Statistics and “funneling”. *File type funneling* is a quantitative measure concerning the relationship between the number of files any process opens and the number of their types. Applications handle (both as input and output) files belonging to a well defined set of types. Therefore, a process which opens many files of an unexpectedly high number of different types may be worth of growing attention.

A simple (though not the only) definition for funneling is due to Scaife et al. (2016):

File System Traversal. Ransomware may thoroughly explore the victim’s file system looking for target files. To this end, the malicious process uses directory-related APIs and system calls with a unusual frequency.

Files Access Statistics. Usually, benign processes open a limited subset of the files stored in the whole file system. In malicious processes, the number of accessed files is generally larger and grows with the execution time.

Further Malice Indicators. Further malice indicators, not only related to file system operation, have been proposed to be “aggregated” with the formers, in order to mitigate false positives and to improve the classification process.

The system proposed in Song et al. (2016) leverages processor, and memory usage statistics, in order to enrich information obtained with file access monitoring. PayBreak Kolodenker et al. (2017a) is a system which tells apart suspicious processes by searching the memory for evidences of in-progress execution of encryption algorithms.

Hardware performance counters in ARM based devices (running the Android OS) are the basis of the proposal presented in Demme et al. (2013).

Power consumption promises to be a viable malice indicator. Indeed, peaks in the amount of energy used by a process can reveal that potential harmful activities (*e.g.*, compulsory data encryption and file system access) are in progress. Current operating systems provide the possibility of measuring the power used by a specific process/set of processes Catuogno et al. (2017, 2018) that, in turn, might be used to identify the resources such processes use.

T. J. Richer Richer (2017) investigates the possibility of measuring the entropy of obfuscated network messages, in order to tell apart botnet (and, arguably, ransomware) C&C communications.

3.1 Discussion

Typically, ransomware are assumed to follow a “greedy” strategy - *i.e.* once in execution, they attempt to seek and encrypt the largest number of user files in the shortest possible time. This is to increase the probability of capturing “precious” documents and to minimize the probability to be detected in the meantime. Therefore, ransomware behaviors may deviate quite soon from that of benign processes,

Table 2: Summary of most widely used indicators of potential ransomware activity.

	ShieldFS	UNVEIL	Redemption	Mbol and Sadighian	DaD	Cryptolock	RansomCare
Per file R/W stats			✓				
Entropy variation in overwrites	✓	✓	✓			✓	✓
Kullback-Liebler				✓			
χ^2					✓		
Similarity						✓	
File moving/removal stats	✓		✓			✓	
Files type modification rate			✓			✓	✓
Accessed file types stats and funneling	✓					✓	
File system traversals	✓		✓				
Files access statistics	✓		✓				

if measured with an adequate set of indicators. Nevertheless, it is possible that benign processes may appear to behave as ransomware if considering single (or few) indicators. Typical examples of such processes are zip-archivers which usually open plenty of files (of multiple types) and feature high differences between read and written data entropy. However, it must be pointed out that such “benign misbehaviors” occur only occasionally and temporarily, while within ransomware, they happen on regular basis.

To cope with this aspect, some indicators come with a *threshold* which can be chosen according different policies. Whenever the score of an indicator exceeds the threshold, the event is notified to the component which makes decision about suspected processes (classifier). Optionally, at every event occurrence, a counter (related to the event itself) is incremented and only when in turn the counter exceeds its own threshold, the decision-maker component is notified.

In order to improve the classification process (and in particular for the sake of reducing the rate of false positives), indicators can be weighted on the basis of experimental findings and incremental refinements as well as hierarchies amongst indicator can be introduced (Scaife et al., 2016).

The set of thresholds and weights, implicitly defines the edge between ransomware and benign software. The point is that several researches have pointed out that ransomware might implement strategies through which they can carry out their job without triggering any event or fatally delaying their discovery.

Indicator Evasion. techniques have been widely investigated in literature. Ransomware might circumvent frequency based indicators (both content-based and behavioral) by slowing down the pace of their file accesses; encrypting files in multiple rounds (Continella et al., 2016) or by encrypting only specific parts of each file instead of encrypting it entirely (Kharraz et al., 2016).

Content-based indicators pose several issues. Firstly, these metrics looks rather prone to high rates of false positives if related to processes performing files compression, (legitimate) files encryption and compressed images manipulation (Gaspari et al., 2021, Pont et al., 2020).

Secondly, such indicators may be eluded by proceeding to encrypt/overwrite files content following rather simple heuristics. For example, a malicious process could avoid entropy related indicators by interleaving cyphertext overwrites with low-entropy paddings when encrypting files (Kharraz and Kirda, 2017, Mbol et al., 2016, Palisse et al., 2017).

A suitable solution to this problem is “aggregating” indicators. Indeed, it looks quite unlikely that a malicious process is able to mimic benign processes according all considered indicators. To this end the CryptoLock (Scaife et al., 2016) system features a *union* indicator which is triggered by the simultaneous occurrence of the events related to its primary indicators. Experiments show that this solution helps to lower false positives.

Multi-process ransomware are still considered an insidious threat. In facts, such viruses distribute their different file re-encryption subtasks among multiple agents. Each agent is likely to avoid to be detected due to its individual subtask may not be classified as

malicious activity (Gaspari et al., 2020, Palisse et al., 2017)

4 FILE SYSTEM-CENTRIC RANSOMWARE DETECTION

This section briefly reports recent ransomware detection systems that use strategies described above. Amongst seminal papers concerning file system-centric ransomware detection, we mention the study due Kharraz et al. (2015) and the UNVEIL system (Kharraz et al., 2016). In particular, UNVEIL leverages a malice indicator based on entropy. The Redemption (Kharraz and Kirda, 2017) system, offers a comprehensive indicators-based approach, and investigates the possible solutions to aggregate multiple indicators into a single *malice score*.

ShieldFS (Continella et al., 2016) features multiple behavioral indicators in order to detect ransomware activity. Indicators include: entropy, fraction of accessed filetypes, fraction of accessed files (r/w). A versioned filesystem is used to reverse possible malicious files encryption. RDS3 (Subedi et al., 2017) is similar to ShieldFS though it uses backups data in spare/unused storage space that is supposed ransomware cannot touch. The data retention mechanism is put in charge to a trusted virtual machine.

CryptoLock (Scaife et al., 2016) follows a similar approach. In order to reduce the occurrence of false positives and to quicken ransomware detection, authors divides indicators in two categories: primary and secondary. Primary indicators include: per-process file-type changing rate, Shannon entropy and *similarity measurement* (Roussev, 2010). Secondary indicators are file deletion per process and funneling. A *union indicator*, related to the simultaneous rise of the primary indicators is also featured.

Mbol et al. (2016) leverage the Kullback-Liebler divergence (Kullback and Leibler, 1951) to realize whether a process is turning a structured file data (*e.g.*, a jpeg image) into an encrypted file.

R2D2 (Gutierrez et al., 2018) addresses the detection of so called “wiper” ransomware. The system, which is built on top of a virtualization infrastructure, leverages VMI to detect the execution of secure deletion algorithms on the target file system.

FlashGuard (Huang et al., 2017) aims at preventing/delaying clear data deletion by leveraging SSD hard disks properties. In facts, SSDs do not directly overwrite chunks of data in their final destination (due to a certain delete latency). FlashGuard, operates at firmware level making possible to retain such out-of-place temporary copies as “backup” of overwritten

data.

The DaD (Data Aware Defense) (Palisse et al., 2017) system monitors processes’ file activity at run-time and measures variations in data distribution using a metric based on χ^2 goodness-of-fit test (Cochran, 1952) instead of Shannon entropy. In order to validate the achieved results, authors also propose a full-featured test environment *Malware-o-Matic* which actually is suitable for application in the testbed scenario. RansomCare (Faghihi and Zulker-nine, 2021) measures the extent of changes in the files structure and in the files content entropy. A malice score is computed for each Application. Whenever the score reaches the configured Anomalous Data Limit (ADL) the system stops the application and warns the user.

Honeypots can be used to improve ransomware detection as in DcyFS (Kohlbrenner et al., 2017) and R-Locker (Gómez-Hernández et al., 2018). The approach is the following. Decoy files (whose content is assumed to never change) are disseminated throughout the filesystem and an agent periodically verifies whether their content have been changed since the previous check. Changes in decoy files reveals that unauthorized file system activity is in progress.

5 DATASETS

Having rich and up-to-date ransomware collections (datasets) is a central need in developing any ransomware detection system. Indeed, datasets are essential both for the sake of detectors tuning/training and for performance and accuracy evaluation.

Nowadays, several initiatives have arisen for the purpose of gathering and making available samples of malware through on-line repositories such as VirusTotal (Chronicle, 2021), VirusShare (Corvus Forensics, 2021) and Hybrid-Analysis (Hybrid Analysis GmbH, 2018). In such services, users can (1) contribute to the repository by submitting captured malware along with any available information concerning its behavior and origin; (2) submit any suspect executable in order to have it analyzed and (3) query the repository for any malware according its name, classification, fingerprint and so on. In some cases, malware repositories provide applications and API (*e.g.*, VxAPI GmbH (2018)) to enable registered users to automate the interaction with the database and to handle high numbers of queries.

Several research labs have made available the datasets built while developing their projects, for the sake of the repeatability and reproducibility of achieved results. Authors of ShieldFS Continella

Table 3: Experiments setup: testbed filesystem composition, ransomware samples, measured performance.

System	testbed	Act. samples	TP	FP	FN
CryptoLock	Agrawal et al. (2007), Hicks et al. (2008)	492	93%	<i>n/a</i>	<i>n/a</i>
ShieldFS	11 workstations held by volunteers for “several weeks”	305	97.70%	0	0
DAD	DigitalCorpora (Digital Corpora Initiative, 2009)	798	99.37%	<i>n/a</i>	0.62%
R2D2	GovDoc (Garfinkel et al., 2009)	<i>n/a</i>	99.80%	0.20%	0.20%
Redemption	5 workstations held by volunteers for one week	1174	100.0%	0.8%	<i>n/a</i>
UNVEIL	<i>n/a</i> (“typical FS layout”)	2121	96.3%	0	<i>n/a</i>
RansomCare	Agrawal et al. (2007), Hicks et al. (2008)	2389	98.38%	0.0049%	<i>n/a</i>

et al. (2016) disclosed the collection of ransomware samples, the detailed description the testbed used in their experiments along with all produced IRP log-files (Continella et al., 2018). Datasets and further information about experimental results have been released in Andronio et al. (2015) for the Heldroid project and in and in Sgandurra et al. (2016) for the EldeRan project. Another example is the Pay-Break (Kolodenker et al., 2017a) team that has made available the RADDAR toolkit (Kolodenker et al., 2017b).

6 CONCLUSION

We presented a survey on some existing ransomware detection systems based on the measurement of quantitative indicators representing the activity each process performs on the file system. This approach has proven to be quite effective and promising. In this paper we put the focus on the strategies in choosing and interpreting the indicators set up in each system, investigating their impact on effectiveness and performance. For each considered indicator we discussed how to face problems like containing false positive and evasion attempts.

In conclusion we would put forth some point that, in our opinion, are worth of further reflections. First, a common indicators specification is still due. In some cases, the same statistics are computed differently as well as the same indicator has rather different definitions (e.g., funneling). Second, testbed filesystems used for performance evaluation are largely different. This makes the different systems quite hard to be compared as this factor remarkably affect performance measurements. Finally, common guidelines for performance measurements are probably due. Different projects leverage different benchmarking tools, each measuring different things.

In this work, we pay a great attention to the definition of malice indicators as we believe that, due to its characteristics, ransomware can be effectively contained by correctly interpreting the track it leaves on the file system rather than attempting to recognize its

executable code.

Nevertheless, we left aside aspects related to the classification of suspect processes. To this end, the overwhelming majority of ransomware detectors currently on the shelf leverages machine learning techniques. Different techniques have different performance in different scenarios (*operational vs testbed*). This may depend from different factors. Moreover, the quality of datasets and samples is very important as well. The availability of many datasets and the growing diffusion of on-line cooperative repositories (such as VirusShare) has boosted the development of new interesting solutions. However, the risk that adversaries could submit misleading datasets, in order to affect the detectors training and testing process, is concrete and has to be coped with. We plan to extend our investigation in both this direction.

REFERENCES

- Agrawal, N., Bolosky, W. J., Douceur, J. R., and Lorch, J. R. (2007). A five-year study of file-system metadata. *ACM Transactions on Storage (TOS)*, 3(3):9.
- Andronio, N., Zanero, S., and Maggi, F. (2015). HelDroid: Dissecting and detecting mobile ransomware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 382–404. Springer.
- Bordoni, L., Conti, M., and Spolaor, R. (2017). Mirage: Toward a stealthier and modular malware analysis sandbox for android. In *European Symposium on Research in Computer Security*, pages 278–296. Springer.
- Bulazel, A. and Yener, B. (2017). A survey on automated dynamic malware analysis evasion and counter-evasion: Pc, mobile, and web. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium*, page 2. ACM.
- Catuogno, L. and Galdi, C. (2016). On the evaluation of security properties of containerized systems. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pages 69–76.
- Catuogno, L., Galdi, C., and Pasquino, N. (2017). Measuring the effectiveness of containerization to prevent power draining attacks. In *2017 IEEE International*

- Workshop on Measurement and Networking (M N)*, pages 1–6.
- Catuogno, L., Galdi, C., and Pasquino, N. (2018). An effective methodology for measuring software resource usage. *IEEE Transactions on Instrumentation and Measurement*, 67(10):2487–2494.
- Chen, P. M. and Noble, B. D. (2001). When virtual is better than real [operating system relocation to virtual machines]. In *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, pages 133–138.
- Chronicle (2021). Virustotal community. <https://www.virustotal.com>.
- Claudio Guarnieri, e. a. (2011). Cuckoo sandbox web page. <https://cuckoosandbox.org>.
- Cochran, W. G. (1952). The χ^2 test of goodness of fit. *The Annals of Mathematical Statistics*, pages 315–345.
- Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barengi, A., Zanero, S., and Maggi, F. (2016). ShieldFS: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347. ACM.
- Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barengi, A., Zanero, S., and Maggi, F. (2018). Shieldfs website. <http://shieldfs.necst.it/>.
- Corvus Forensics (2021). Virusshare repository. <https://virusshare.com>.
- Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., and Stolfo, S. (2013). On the feasibility of online malware detection with performance counters. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 559–570. ACM.
- Digital Corpora Initiative (2009). Corpora. <http://digitalcorporas.org/corpora/>.
- Don Capps et al. (2002). IOZone file system benchmark. <http://www.iozone.org/>.
- Faghihi, F. and Zulkernine, M. (2021). Ransomcare: Data-centric detection and mitigation against smartphone crypto-ransomware. *Computer Networks*, 191:108011.
- Garfinkel, S., Farrell, P., Roussev, V., and Dinolt, G. (2009). Bringing science to digital forensics with standardized forensic corpora. *digital investigation*, 6:S2–S11.
- Garfinkel, T. and Rosenblum, M. (2003). A virtual machine introspection based architecture for intrusion detection. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. The Internet Society.
- Gaspari, F. D., Hitaj, D., Pagnotta, G., Carli, L. D., and Mancini, L. V. (2020). The naked sun: Malicious cooperation between benign-looking processes. In Conti, M., Zhou, J., Casalicchio, E., and Spognardi, A., editors, *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part II*, volume 12147 of *Lecture Notes in Computer Science*, pages 254–274. Springer.
- Gaspari, F. D., Hitaj, D., Pagnotta, G., Carli, L. D., and Mancini, L. V. (2021). Reliable detection of compressed and encrypted data. *CoRR*, abs/2103.17059.
- GmbH, H. A. (2018). A generic interface and CLI for all endpoints of the Falcon Sandbox API. <https://github.com/PayloadSecurity/VxAPI>.
- Gómez-Hernández, J., Álvarez-González, L., and García-Teodoro, P. (2018). R-Locker: Thwarting ransomware action through a honeyfile-based approach. *Computers & Security*, 73:389–398.
- Google Inc. (201x). Android fileobserver. <https://developer.android.com/reference/android/os/FileObserver>.
- Gutierrez, C. N., Spafford, E. H., Bagchi, S., and Yurek, T. (2018). Reactive redundancy for data destruction protection (R2D2). *Computers & Security*.
- Hicks, B. J., Dong, A., Palmer, R., and Mcalpine, H. C. (2008). Organizing and managing personal electronic files: A mechanical engineer’s perspective. *ACM Transactions on Information Systems (TOIS)*, 26(4):23.
- Huang, J., Xu, J., Xing, X., Liu, P., and Qureshi, M. K. (2017). Flashguard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2231–2244. ACM.
- Hybrid Analysis GmbH (2018). Hybrid analysis. <https://www.hybrid-analysis.com>.
- Kharraz, A., Arshad, S., Mulliner, C., Robertson, W., and Kirda, E. (2016). UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772. USENIX Association.
- Kharraz, A. and Kirda, E. (2017). Redemption: Real-time protection against ransomware at end-hosts. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 98–119. Springer.
- Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., and Kirda, E. (2015). 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.
- Kohlbrenner, A., Araujo, F., Taylor, T., and Stoecklin, M. P. (2017). POSTER: Hidden in plain sight: A filesystem for data integrity and confidentiality. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2523–2525. ACM.
- Kolodenker, E., Koch, W., Stringhini, G., and Egele, M. (2017a). Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 599–611. ACM.
- Kolodenker, E., William, K., Gianluca, S., and Manuel, E. (2017b). Real-time Automation to Discover, Detect and Alert of Ransomware (RADDAR). <https://github.com/BUseclab/raddar>.
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.
- Le, T. (2015). A recommended framework for anomaly intrusion detection system (ids). In *GI-Jahrestagung*, pages 1829–1840.

- Lengyel, T. K., Maresca, S., Payne, B. D., Webster, G. D., Vogl, S., and Kiayias, A. (2014). Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 386–395. ACM.
- Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.
- Ma, W., Duan, P., Liu, S., Gu, G., and Liu, J.-C. (2012). Shadow attacks: automatically evading system-call-behavior based malware detection. *Journal in Computer Virology*, 8(1):1–13.
- Mbol, F., Robert, J.-M., and Sadighian, A. (2016). An efficient approach to detect torrentlocker ransomware in computer systems. In *International Conference on Cryptology and Network Security*, pages 532–541. Springer.
- Microsoft Inc. (2014). File system minifilter drivers. <https://docs.microsoft.com/en-gb/windows-hardware/drivers/ifs/file-system-minifilter-drivers>.
- Milosevic, J., Sklavos, N., and Koutsikou, K. (2016). Malware in iot software and hardware.
- More, A. and Tapaswi, S. (2014). Virtual machine introspection: towards bridging the semantic gap. *Journal of Cloud Computing*, 3(1):16.
- Oberheide, J., Bailey, M., and Jahanian, F. (2009). Poly-Pack: an automated online packing service for optimal antivirus evasion. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, pages 9–9. USENIX Association.
- Palisse, A., Durand, A., Le Bouder, H., Le Guernic, C., and Lanet, J.-L. (2017). Data aware defense (dad): Towards a generic and practical ransomware countermeasure. In *Nordic Conference on Secure IT Systems*, pages 192–208. Springer.
- Payne, B. D., Maresca, S., Lengyel, T. K., and Saba, A. (2015). LibVMI Github repository. <http://libvmi.com/>.
- Pont, J., Arief, B., and Hernandez-Castro, J. (2020). Why current statistical approaches to ransomware detection fail. In Susilo, W., Deng, R. H., Guo, F., Li, Y., and Intan, R., editors, *Information Security - 23rd International Conference, ISC 2020, Bali, Indonesia, December 16-18, 2020, Proceedings*, volume 12472 of *Lecture Notes in Computer Science*, pages 199–216. Springer.
- Richer, T. J. (2017). Entropy-based detection of botnet command and control. In *Proceedings of the Australasian Computer Science Week Multiconference*, page 75. ACM.
- Roussev, V. (2010). Data fingerprinting with similarity digests. In *IFIP International Conference on Digital Forensics*, pages 207–226. Springer.
- Scaife, N., Carter, H., Traynor, P., and Butler, K. R. (2016). Cryptolock (and drop it): stopping ransomware attacks on user data. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 303–312. IEEE.
- Sgandurra, D., Muñoz-González, L., Mohsen, R., and Lupu, E. C. (2016). Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*.
- Song, S., Kim, B., and Lee, S. (2016). The effective ransomware prevention technique using process monitoring on android platform. *Mobile Information Systems*, 2016.
- Subedi, K. P., Budhathoki, D. R., Chen, B., and Dasgupta, D. (2017). Rds3: Ransomware defense strategy by using stealthily spare space. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE.
- Underwriters Labs LLC (2013). Pemark 8. <https://benchmarks.ul.com/>.
- Viswanathan, A., Tan, K., and Neuman, C. (2013). Deconstructing the assessment of anomaly-based intrusion detectors. In *International Workshop on Recent Advances in Intrusion Detection*, pages 286–306. Springer.
- Young, A. and Yung, M. (1996). Cryptovirology: extortion-based security threats and countermeasures. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 129–140.