Data Aware Defense (DaD): Towards a Generic and Practical Ransomware Countermeasure

Aurélien Palisse, Antoine Durand, Hélène Le Bouder, Colas Le Guernic, Jean-Louis Lanet

Data Aware Defense (DaD): Towards a Generic and Practical Ransomware Countermeasure

Aurélien Palisse[1] ✉, Antoine Durand[2], Hélène Le Bouder[3], Colas Le Guernic[1,4], and Jean-Louis Lanet[1]

[1] INRIA, Campus de Beaulieu, 263 Avenue Général Leclerc, Rennes - France
aurelien.palisse@inria.fr
[2] ENSEIRB - MATMECA, 1 Avenue du Dr Albert Schweitzer, Talence - France
[3] IMT Atlantique, 2 Rue de la Châtaigneraie, Cesson Sévigné - France
[4] DGA - Maîtrise de l'Information, Route de Laillé, Bruz - France

Abstract. We present the *Malware - O - Matic* analysis platform and the *Data Aware Defense* ransomware countermeasure based on real time data gathering with as little impact as possible on system performance. Our solution monitors (and blocks if necessary) file system activity of all userland threads with new indicators of compromise. We successfully detect 99.37% of our 798 active ransomware samples with at most 70 MB lost per sample's thread in 90% of cases, or less than 7 MB in 70% of cases. By a careful analysis of the few false negatives we show that some ransomware authors are specifically trying to hide ongoing encryption. We used free (as in free beer) de facto industry standard benchmarks to evaluate the impact of our solution and enable fair comparisons. In all but the most demanding tests the impact is marginal.

1 Introduction

Ransomware is a type of malware that prevents legitimate users from accessing their machine or files and demands a payment for restoring the functionalities of the infected computer. There are two classes of ransomware: the "simple lockers", which block the usage of the computer, and "cryptors", that encrypt files on the computer. In the case of encryption-based ransomware, the user data can only be restored with the secret key(s) used during the attack.[5]

This class of malware has existed for a few decades [32], but the number of attacks has increased drastically over the past couple of years [25]. The latest notable examples are the WannaCry and the Nopetya attacks. However, recent findings suggest that Nopetya is a wiper with ransomware-like appearance. Microsoft is concerned by the ransomware threat and plans to add a controlled folder access feature in the next operating system update [3].

Recent advances have contributed in the current proliferation of ransomware. Command and control (C&C) servers can be protected through the use of do-

[5] Usually the encryption keys are themselves encrypted with an asymmetric cryptosystem, the ransom must be paid in order to get the corresponding private key.

main generation algorithms or The Onion Router (TOR) network. Popular applications have been diverted from their legitimate usage: Imgur [27], Twitter API [1], and Telegram Bot API [12] have been used to implement a C&C. Bitcoin and other cryptocurrencies (e.g, Zcash, Ethereum) facilitate ransom processing and handling. Similarly to most modern malware, current ransomware, hinder detection and analysis, through packing, virtualization, Windows Management Instrumentation (WMI) queries or obfuscated API calls. On March 28, Trend Micro discovered a new technique used by the Cerber family to evade static machine learning solutions [28]. A generic efficient defense mechanism seems challenging, but encryption-based ransomware have a clear common semantic that can be taken advantage of: they encrypt user data.

Similarly to recent approaches [5, 13, 14, 23], reporting satisfying detection rates, we propose to monitor file activity. Since it has already been proven a valid approach in terms of detection, our main goal in this paper is to show that a good detection rate can be achieved with little to no impact on system performances. To this end we limit our monitoring to a minimum. In order to reduce the impact on detection with a low rate of false positive, we use the chi-square goodness-of-fit test instead of Shannon entropy (i.e, sensitive to compressed chunks of data [17]). We also achieve system completeness and fine granularity by monitoring the whole file system for all userland threads. In order to evaluate our prototype implementation, *Data Aware Defense* (DaD), under realistic conditions, we developed a bare-metal analysis platform, *Malware - O - Matic* (MoM), and ran it on a large and heterogeneous (compared to the litterature) live ransomware collection. We used de facto industry standard benchmarks to get a pertinent and reproducible assessment of the performance penalties.

Related work are presented in section 2. Section 3 compares different statistical tests and their effectiveness to detect encryption. DaD, our ransomware countermeasure is introduced in section 4, a significant effort is made to tackle the performance bottleneck. We evaluate its impact on the protected system performance in section 5 and its effectiveness in section 6 together with a description of our bare-metal automated malware analysis platform, MoM, and a discussion on our findings. Section 7 concludes the paper.

2 Related Work

Detecting malware is of prime importance. The main deployed approaches are either static pattern-based signatures, or behavioral signatures dynamically checked in a sandboxed environment. Unfortunately they rarely cover new malware or even new variants of known malware. The challenge is to design fast detection schemes that cover many samples with no false positives.

The subfield of ransomware countermeasures is relatively young. On top of classical malware approaches, one can rely on the specific common behavior of ransomware: they encrypt the victim's files. Dynamic solutions found in the literature are divided in two parts as suggested by Kharraz et al. [14]: cryptographic primitives hooks (i.e, user space) and low level disk activity monitoring (i.e, ker-

nel space). One targets the cryptographic primitives as an essential gateway for ransomware whereas the second focuses on the system consequences.

In 1996 Young et al. [33] implemented a proof of concept "cryptoviral extortion" based on the Microsoft's Crytographic API (CryptoAPI). Nowadays a significant number of ransomware do indeed use the CryptoAPI to perform file encryption. This API enables the use of specific cryptographic providers, Palisse et al. [20] implemented their own and forced its use to get a trace of all cryptographic operations. PayBreak [15] live solution makes use of dynamic and static cryptographic hooks. A key escrow system is implemented and allows complete file recovery. However only symmetric-key encryption is considered and some obfuscation techniques will defeat the hooks, according to the authors. Moreover the static hooks need a prior knowledge of the libraries. Both papers suffer from two critical limitations: any ransomware that will embed its own cryptographic primitives bypass the solution (e.g, AES-NI [22]) and nothing prevents the ransomware to detect the hooks or the redirection.

Other approaches, like ours, focus on disk activity thanks to a file system driver. UNVEIL [13] detects ransomware by computing the increase, in term of Shannon entropy, between data read and written to disk by the same process and is also able to detect desktop locker, a benign type of ransomware. CryptoDrop [23] relies on entropy too, but also takes into account file type changes and a file similarity score. ShieldFS [5] applies machine learning to the disk activity. Features are selected from millions of disk I/O requests gathered from normal usage and ransomware attacks. The solution is composed of three drivers. One in charge of file recovery: for each write and renaming operations the corresponding file is backed up. The second detects cryptographic materials embedded in processes. Finally the third performs detection thanks to random forests mixed with incremental models which takes into account the short and long life of a process.

The three report good detection rate, over 96%, and almost no false positive on their respective data sets. Concerning the impact on system performance UNVEIL and CryptoDrop give little information. The former is presented as an analysis tool and not a live solution, the authors of the later "believe that with future optimizations, CryptoDrop can be run on a live system with a small overhead." and report a 9ms overhead per write operation without specifying their evaluation procedure. There are more information on ShieldFS performance overhead. The time taken to open then read, or open the write, files of increasing size (from 1KB to 128MB) is measured. Using hard disk drive they get a 180% to 380% overhead if files need to be backed up or 30% to 90% if not. A "typical" overhead is also reported, unfortunately the evaluation procedure is unclear and hardly reproducible: they extrapolated a typical overhead from IRP logs taken from five users, resulting in an average estimated overhead of 26%.

Dynamic monitoring of file system activity seems to be the most promising approach to defend against ransomware. Indeed, the Intel AES-NI instructions defeat the approaches centered on the cryptographic libraries proposed in [15,20]. The remaining challenge lies in an efficient implementation on a live system.

One approach would be to limit the level of monitoring and focus on a single efficient distinguisher, at least at first, only suspicious threads need to be closely monitored.

3 Statistical Tests for Ransomware Attacks Detection

Ransomware involve a large number of ciphertext going through the file system; to detect such behavior, statistical tests can be used. The main idea is that ciphertext content distribution should be uniform. A one sample goodness of fit (GOF) test measures how close an information source is to a theoretical probability distribution function, also known as the "model". In practice, one data set F is compared to a known distribution function G and disproves or not the null hypothesis $H_0 : \forall x,\, F(x) = G(x)$.

We do not prove that both data sets come from a single distribution function but rather that there is no significant difference between them. The objective is to figure out which indicators of compromise is the most relevant to detect ransomware attacks in real time.

Shannon Entropy It is a measure of the uncertainty of a random variable. Lots of disorder raise high entropy and structured data low entropy. The entropy of X a discrete random variable from the alphabet $\Omega = \{x_1, x_2, \ldots, x_n\}$ with the probability distribution function $p(x_i)$ at x_i is: $H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$.

Chi-Square The chi-square goodness-of-fit test (χ^2) is a test of distributional accuracy, it measures how closely a set of numbers follows a particular distribution. It is a non-parametric statistical test, meaning that no assumption is done on the samples distribution. The observed sequence of data is considered as discrete and arranged in a frequency histogram $[\![0; 255]\!]$ with the degree of freedom v equals to 255 (i.e, number of possible outcome minus one). Suppose that N_i is the number of hit observed for the bin i, and n_i is the expected number according to a known distribution function. The formula for calculating the one-sided χ^2 test is:

$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i} \tag{1}$$

A large value indicates that the null hypothesis is not likely verified, the N_i can not be drawn from the n_i. The significance level of the test denoted α_{TW}, is the probability of rejecting the null hypothesis when it is true. Traditionally, experimenters have used the 0.05 level (e.g, biology), thus we choose the same. The observed test statistic is compared to a boundary value, called the critical value, uniquely determined from the degree of freedom (or equivalently the size of the alphabet) and the desired significance level. If the χ^2 result is more extreme than the critical value [10], the null hypothesis is rejected.

Discussion The robustness of the tests need to be checked against real world conditions (i.e, small and large samples). Previous papers [5, 13, 23], use the plug-in method (i.e, discrete symbols in histogram bins) to estimate the Shannon entropy. Nevertheless a study on TorrentLocker [17] shows that the Shannon entropy is not a good distinguisher especially with respect to JPEG compression[6]. Achieving encryption detection on compressed files that already have high entropy is a non-trivial task. The χ^2 test on the contrary can distinguish randomness (or encryption) from some compression schemes and is thus a more relevant statistic as shown in appendix tables 4 and 5. For that reason we use it to detect suspicious behaviors in the next sections. No extensive study of a ransomware solution embedding the χ^2 has been presented earlier, but this statistic is already used in numerous applications [6, 31].

A practical issue remains, like all statistical tests, the χ^2 test is not accurate on small samples. A good practice is to have at least five elements in each bins of the histogram, but we can reasonably think that this will not happen plenty of time. In this case, the test statistic will only reflect the small magnitude of the expected frequencies. To fix this problem we made the choice that the solution favor false positives over false negatives, by also computing the χ^2 test for small data. Moreover, the number of bytes involved in the computation is limited to 10,000. Indeed, if we do not set a limit, a zip bomb [11] can be used to crash or slow down our solution.

4 Towards a Generic and Practical Ransomware Countermeasure

In this section the architecture of *Data Aware Defense* is detailed, a file system driver for Microsoft Windows. An important part of the contribution is the usability of the solution, furthermore it can be used against zero-day ransomware.

4.1 File System Activity Monitoring

Windows, as most modern operating systems, splits its memory in several regions with different privilege requirements. The kernel mode (ring 0) has a high privilege level and is responsible, among other things, for managing disk operations.

Standard applications run in userland (ring 3), they are much less privileged and cannot perform disk operations directly. Instead they call the corresponding service from the Windows kernel, let the kernel manage the privileged operations, before safely returning in the userland application code. This separation ensures that no userland code directly manage critical operations in the system. To complete this security feature the 64-bit versions of the OS requires all kernel mode code to be signed by Microsoft to be accepted.

[6] They use the Kullback-Liebler divergence instead but do not introduce an implementation.

Up to now, to the best of our knowledge ransomware live in userland. That is why a countermeasure in kernel space can not be tampered with by malicious code and is fully transparent. Users interactions with files are ultimately mapped to operations in the kernel. On top of this stack stand the I/O manager and at the bottom the file system driver (ntfs.sys). Microsoft offers a file system drivers framework [18] that allows third party developers to add functionalities between the two previous layers. Such component is called a file system minifilter driver and it is managed by the filter manager. The position of each minifilter driver in the I/O stack is defined by its "altitude". A minifilter driver that performs full disk encryption is below an anti-virus filter and thus avoid false positive detection of ransomware-like behavior. A minifilter driver can inspect all the operations that target the disks, regardless of whether the requested operation is an I/O request packet (IRP) or a fast I/O. In this context we are able to monitor write, read operations and so on. However a clever usage of such functionalities has to be done, otherwise a significant performance penalty occurs [23] and the solution can not be deployed in real world. Our file system minifilter driver has been extensively tested on Windows 7 and 10, for the 32 and 64-bit versions and follows the Windows Driver Model (WDM).

4.2 Implementation Design

In order to catch malicious behaviors efficiently we restrict our monitoring to write and information operations. We want to demonstrate that detecting ransomware behaviors with only two callbacks on the I/O requests is possible, while previous solutions [5, 13, 23] have at least twice as many callbacks. The so-called information operations allow to change various information about a file object, create a hard link, change the file position or the file name. We block all the information operations once a thread is marked as malicious to prevent aggressive files renaming. But without malicious activity, the information operations are always accepted and are not used to construct our compromise indicators. For each intercepted write operation the time spent in the callback function is minimized by collecting only the essential information. We are only interested in the content of the buffer that is passed through the file system stack, its size, offset, the corresponding absolute file name, process name, process id and thread id. This information is then copied to nonpaged memory and passed over to a new dedicated thread that is not part of the file system stack. Only the indispensable data is copied, not the thread context coming from the operation. As a consequence, write operations are immediately authorized to go through. Such features allow us to monitor all file system trees without excluding some assumed trusted threads. So far, we are the first to inspect the I/O operations with a thread granularity. It is a significant improvement in case of malicious code injection into a benign process. The Cerber ransomware already used a particular code injection technique, named process hollowing [28]. All costly computations are deferred to threads running at the lowest kernel priority level. This model solves the time restriction for the statistical computations and the synchronization deadlocks on shared resources.

Once the compromise indicator (e.g, χ^2) is obtained on the corresponding data we update an internal structure related to this thread behavior in nonpaged memory. This structure is stored for each thread of each process. To reduce the memory footprint, our own garbage collector has been implemented. All tracked information in memory can be exported to disk as a JSON file.

4.3 A Single Indicator of Compromise

To build an efficient filter, a sliding median on the last fifty write operations is computed. The goal of this basic statistic is to capture, for each thread, the ongoing file system behavior: it gives us a measure of central tendency. No assumption is made on the I/O patterns observed. This elementary statistic is low cost and does not involve complex calculations. Furthermore by monitoring the whole file system we have more chances to only lose files that did not belong to the user's important paths (e.g, Windows Defender, Python libraries, $Recycle.Bin).

At runtime if the χ^2 median go beyond the threshold, we suspend the corresponding process and collect information for postmortem analysis. We take advantage of the process suspension (i.e, malicious code will not notice) to dump from RAM the Portable Executable (PE) file and the committed pages of memory that belong to this process threads. All the process threads are stopped to ensure consistency of the memory dump, especially in case of self-modifying code. Finally, the thread that triggered the dump is tainted as malicious and all subsequent write and renaming operations are blocked for this particular thread.

Section 3 presents the χ^2 test. The significance level α_{TW} is set to be small (i.e, 0.05). It corresponds to the probability of rejecting the null hypothesis when it is true (i.e, type I error). This parameter also called the "testwise" alpha is relevant for one given hypothesis test. The sliding median is based on fifty consecutive χ^2 tests. Thus, the "experimentwise" alpha is the probability of having one or more errors of type I within the hypothesis tests. The experimentwise error rate can be calculated using the Bonferroni threshold [2] as follows:

$$\alpha_{\text{EW}} = 1 - (1 - \alpha_{\text{TW}})^K \tag{2}$$

K is the number of uncorrelated hypotheses being tested at the α_{TW} level. However, the experimentwise error rate is the same than the testwise error rate when only one hypothesis is tested for a given dataset. Moreover, it is likely that for each thread the write operations are correlated to each others. Therefore, the experimentwise error rate is equal to 0.05.

5 Experiments: Performance Evaluation

An important contribution in this paper is to address the performance penalty. This section shows that our solution is the first live ransomware coutermeasure based on a file system driver able to tackle the performance bottleneck. The workload model is office work. In this configuration the solution is almost

imperceptible. In the following parts, the global impact of the solution on the system is investigated with de facto industry standards software[7]. No formal studies of the in-memory footprint have been conducted, it can be considered negligible on consumer computers (i.e, less than 50 Mb). All test were run under Windows 7 with 4Gb of RAM and a SSD[8]. In order to focus on performance and not interfere with the benchmarks, we deactivated the blocking mechanism of our solution.

5.1 Disk Performance

To precisely measure the driver impact on disk, we used the Windows Performance Toolkit (WPT) [19]. It produces in-depth performance profiles of Windows operating systems. We set the minifilter I/O activity scenario for two hours and a half and obtained 11348 writes that give us $\approx 133ms$ spent in total in the file system driver. Thus, an average of $11.7\mu s$ per operation. During the profiling, normal work activities were simulated, with office suite use, downloads and compression. CryptoLock [23] produces an overhead of $9ms$ per write operation, even without knowing their testing procedure, it is probably safe to say that performance has been significantly enhanced by a factor of a few hundreds.

Additionally a second test based on a free software, CrystalDiskMark [7], has been performed. Two types of write operations are considered: "Sequential" correspond to large contiguous blocks of data and "Random" focus on small (4K) blocks to random locations. As shown in table 1, with the Random 4K tests our solution lead to a significant loss of performance but the resulting bandwidth of 9.5 MB/s is still reasonable. On an hard disk drive (HDD) the seek time and the rotational latency create a bottleneck. Consequently, the Random 4K tests will be bounded by the HDD mechanical limitations (i.e, 2.5 MB/s for the fastest HDD [9]), not the file system driver. On the other hand no performance loss is observed when dealing with big chunks of data. Pushing the limit of the solution on disk access do not create undesirable effects (e.g, freeze).

Table 1: CrystalDiskMark tests (in MB/s) configured with 5 tests passes, file size of 2GiB, random data generation and 3 minutes interval time.

Write test	Solution off	Solution on	Impact
Sequential	136.3	134.2	−1.5%
Sequential $Q32T1$	135.7	135.8	+0.07%
Random $4K$	55.28	9.581	−82.66%
Random $4KQ32T1$	122.0	65.48	−46.32%

[7] We restricted ourselves to free (as in free beer) softwares used to assess performance of personal computers to ensure pertinence and affordable reproducibility.

[8] Windows 7 SP1 6.1.7601, Intel Xeon W3550, NVIDIA Quadro FX 1800, 4Gb DDR3, Intel SSD 120 Go SATA III

[9] `http://hdd.userbenchmark.com/WD-Black-6TB-2015/Rating/3519`

5.2 CPU Performance

Due to the high number of threads that complete asynchronous jobs, the system load needs to be considered. For this we used Geekbench 4 [9] that contains an all in one test with different workload models and PCMark 8 [21] that is recognized as an industry standard benchmark. For more details on the underlying tests performed by the software please refer to their technical reports. We measure initially (without our protection) a Geekbench overall score of 6625 (multi-core) and 5841 after enabling the protection, for a performance loss of 11.83%. Contrary to the previous test, with PCMark the system impact is less palpable. We executed two built-in scenarios, "work" that measures basic work tasks on office machine and "home" that focus more on media capabilities (e.g, gaming, photo, chat). As shown in table 2, only a very small difference occurs (less than 1%).

Table 2: PCMark 8 benchmark score.

Test	Solution off	Solution on	Performance loss
Work conventional 2.0	2859	2845	−0.49%
Home conventional 3.0	2728	2705	−0.84%

5.3 Discussion

Precise comparison with other contributions [5, 23] is difficult: we can not reproduce their evaluation procedure (when they have one) and we were not able to apply our own[10]. Still, with a significant impact only on the most demanding test that is only exhibited when using solid state drives, we can safely assume a significant improvement over previous work and report that DaD is practical.

6 Experiments: Ransomware Detection

In this section, the experiments demonstrate that *Data Aware Defense* (DaD) detects and blocks in real time, most of the ransomware in the collection (i.e, 99.37%) with a low number of bytes lost. Then we discuss about the ransomware-like behaviors that lead to false positives, and finally review two false negatives that bypass the countermeasure using mimicry attacks.

Before delving into the experimental evaluation of our approach and an analysis of ransomware behavior in the next section, let us present our malware analysis platform that was used to conduct those experiments.

6.1 Malware - O - Matic

We designed and built *Malware - O - Matic* (MoM), an automated analysis platform that does not use a virtual machine, while keeping all the main features

[10] We solicited the authors and got a negative answer from [5], and no answer from [23] as of submission.

of a regular analysis framework. Such fully bare-metal platform is built on top of two open source software, Clonezilla [4] and Viper [29], which makes it reproducible. The platform is made of a single master server and several slaves, each one running the analysis loop in parallel. The whole system is on a dedicated network under the supervision of the network autonomous system (AS) and directly connected to the Internet, to emulate a typical home network. The loop itself consists in a few simple steps: setup of the monitoring environment, malware execution, results gathering and storage, cleanup. In the first step the slave download a script from the master, that will act as instructions about how to conduct the next analysis. Once the procedure is completed, the slave sets its next environment and reboot for cleanup. The cleanup process simply consists of flashing a clean disk image onto the slave's drive. So far, MoM is able to analyze up to 360 malware per-day with only 1 server and 5 slaves. The end goal of this platform is to run uninterruptedly and thus automate the analysis of samples.

6.2 Experimental Setup

MoM is used in two distinct modes for the experiments: "passive" or "active". The first one, downloads a sample, executes it and according to a cryptographic hash already determined on the user files, labels the sample as active if the hash changes or discards it. The second mode, evaluates DaD ransomware countermeasure (i.e, file system driver) with the samples marked as actives. With such scenario, once the analysis is complete a set of information (e.g, PE file) is sent to a remote server. To avoid evasion during the analysis, a corpus of files that looks like a plausible user environment is built, thanks to the Digital Corpora corpus [8] and manual additions. In the same way, a set of user interactions is emulated (e.g, mouse, keyboard). Each run is fifteen minutes long. A Windows 7 SP1 32-bit snapshot is chosen as the operating system to be infected, the user is logged in as administrator with the User Account Control (UAC) disabled.

The experiments are based on a long-term collection gathered from August 2016 to March 2017. A VirusShare archive dedicated to ransomware was used in combination with daily crawling on online repositories [16, 30]. Such mixing allows us to have an heterogeneous ransomware collection of 798 active ransomware (i.e, they encrypt the user's files), decomposed in more than twenty families, with numerous singletons. The previous studies due to their virtualized analysis environment were unable to run the Cerber samples[11]. Our dataset is not limited by the anti-virtualization techniques. Samples labeling is achieved through the *Avclass* tool [24]. Detailed information about the collection can be found in appendix, table 3. For each sample, a manual analysis has been performed in accordance with its JSON log file to highlight irrelevant samples, but also false positives and negatives.

[11] PayBreak did, might be samples mislabeling.

6.3 Detection Results

DaD is only interested in the write operations on disk, with a thread granularity and irrespectively of any signature. Such feature makes the solution agnostic which is necessary to tackle zero-day ransomware. The solution successfully detects 99.37% of our ransomware. Solely five circumvent the countermeasure. The following activities are simulated during the evaluation: mouse move, keyboard input and web browser usage. Only one false positive is encountered as seen fig. 1. Up to 238K threads have been monitored during the samples evaluation. A very important point is that DaD's classification error rate is very low: $7.08e-05$.

		Prediction outcome	
		1	**0**
Actual value	**1**	True Positive 1870	False Negative 16
	0	False Positive 1	True Negative 238098

Fig. 1: The confusion matrix related to the suspicious (1) and non suspicious (0) threads monitored by *Data Aware Defense* (DaD) during the samples evaluation.

To assess the effectiveness of DaD, an estimation of the number of bytes lost across the ransomware collection is displayed fig. 2. One can notice that for 70% of the samples' threads, at most 6.5 megabytes (MB) are encrypted, which is acceptable for most of the users. Unfortunately, considering 90% of the collection, 70 MB is lost per-thread in the most extreme scenario. Depending on the user needs, such loss can be tolerated, but it might be unacceptable for businesses. Most of the samples are single threaded, respectively 76.88%.

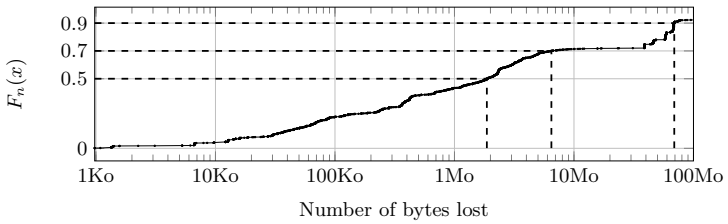

Fig. 2: The cumulative probability of the malicious threads for each thread number of bytes lost.

The detection is affected by the nature of the explored paths, and may be more or less prompt to block a malicious thread. Indeed, the folders with a few number of encrypted files set the sliding median far beyond the detection threshold. The epidemic of ransom notes is to blame. Even when this scenario occurs

the malicious thread is successfully stopped. It demonstrates that monitoring the whole file system makes our solution resilient. Moreover, the compromise indication, a χ^2 sliding median on the last fifty write operations can be circumvented if less than half of the thread activity is dedicated to files encryption.

An important observation that we made, is that different χ^2 "layers" can be distinguished on the disk. Each one corresponding to a specific behavior, such as ransom notes and metadata appended to files. These patterns suggest a criterion to distinguish reversible from non-reversible ransomware. Indeed, metadata appended to files during encryption are visible on the file system and suggests a chance to get the data back (e.g, authors implement the decryption routine). Furthermore, no dissociation into separate threads has been observed for all the three following tasks: files encryption, ransom notes and the metadata.

6.4 Ransomware-Like Applications

Experimental results point that the solution is effective to stop infection but we did not discuss about limitations, in particular false positives and negatives. The primary purpose was to design DaD as an practical and efficient first line of defense against the ransomware.

False Positives Looking at the χ^2 sliding median with a significance level, α_{EW} of 0.05, allows us to eliminate a significant number of false positives among most of the traditional applications. DaD monitors about dozens of processes and hundreds of thread while an user interacts with its machine. Few applications are blocked, in such cases, it disables a particular task (e.g, update), not the entire process. Moreover only very specific applications are able to obtain malicious file system behavior: files compression or encryption, secure files deletion, browsers startup and so on as shown fig. 3.

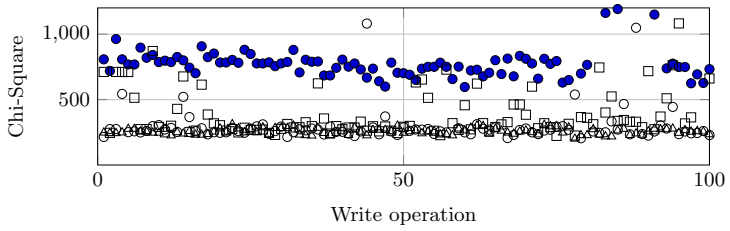

Fig. 3: χ^2 of the 100 first write operation of Mogrify (•), 7-zip (□), GPG4Win (△), and μTorrent (○).

Indeed, the solution is not yet able to distinguish compression from encryption and a false positive is raised with 7-Zip, GPG4Win, or μTorrent. Still, as mentioned in section 3 we can distinguish JPEG compression. The χ^2 statistic corresponding to the images rewritten by the Mogrify software is far away from the critical value (i.e, 293.24). In any cases, a major observation can be made: only a single "layer" is present. The χ^2 statistic alone is not sufficient to avoid

false positives. The ransomware business model is based on extortion, in order to be paid, they need to make the ransom notes as visible as possible. Future works should focus on this idea.

False Negatives The *Data Aware Defense* ransomware countermeasure focus on very specific activities on disk that belong to ransomware behavior but not exclusively. The underlying mechanics that comes with the ransomware to date is well known and documented: they encrypt files. However, the ransomware industry is very prolific and no one is immune to more stealthy behaviors, we are faced with an arms race. For example, the specific problem of boot sectors encryption (e.g, master boot record) is not addressed in this paper, a solution is proposed by the Talos Group [26]. In addition, as outlined in Mbol et al. [17], if an encryption algorithm preserving the distribution of the original files is used, it will evade the solution because randomness is the root of the detection. The ransomware which interleave malicious write operations with loops of unnecessary or redundant operations that look non random will go through DaD, as shown fig. 4. Prior to block a malicious thread, DaD need to have a windows on the last fifty write operations. A multi-threaded ransomware where each file is encrypted by a unique thread can exploit this limitation. Finally, a kernel exploit is a potential breach that ransomware might use to unload DaD and more, in this scenario, the system is completely compromise.

Five ransomware samples among the collection (i.e, 0.62%) bypass the solution. Three Xorist samples used weak encryption algorithms (e.g, Tiny Encryption Algorithm). The last two samples behaviors are different than all we have previously observed. Figure 4, illustrates such statements.

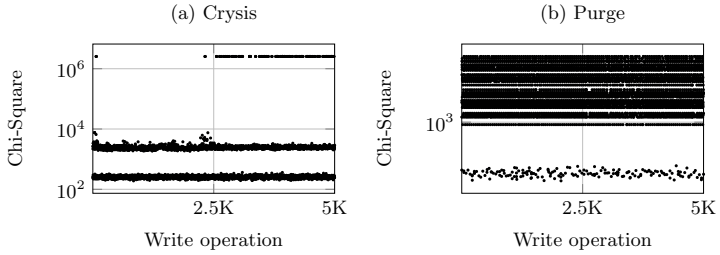

Fig. 4: Two false negatives samples: Crysis and Purge. Both interleave each malicious write operation with numerous garbage operations.

The Crysis sample does not write ransom notes, and after approximately 3,000 write operations begin to perform large write operations of 2^{18} bytes. Such operations with a zeroed buffer are repeated multiples times. The Purge sample hide his malicious behavior through a set of heterogeneous write operations. For each encrypted files, the ransom note is rewritten with chunks of 128 bytes. In both cases, the sliding median is inefficient to detect the underlying encryption process, the willingness of the ransomware authors is to hide the primary purpose of the application behind useless operations (i.e, mimicry attack). The ransomware can not be seen any more like a simplistic version of malware,

in the future they will pretend to do something else than just encrypting files, which was not the case so far to the best of our knowledge.

7 Conclusions

The *Data Aware Defense* is based on file system monitoring and no assumptions is done concerning the malicious I/O patterns. In addition we achieved a thread granularity control on the system and do not restrict files protection on a particular folder. The χ^2 test by its own can replace the Shannon entropy and catch up some of its weaknesses. Moreover our countermeasure is efficient and can be deployed on Windows 7/10 machines with a reasonable performance hit, with an average delay of $12\mu s$ per write operation on disk, a few hundred times smaller than previous approaches. Our extensive experiments show that the more sophisticated ransomware already use mimicry attacks. However we successfully detect 99.37% of the samples with at most 70 MB lost per sample's threads in 90% of cases and less than 7 MB in 70% of cases.

These promising results in terms of performance and detection rate were obtained thanks to single simple metric computed for all threads of all processes running on the system, allowing us to track code injection attacks in particular. False positives seem inherent to the approach: we are detecting large write operations of random data. But its speed and low negative rate makes it a good candidate as a first line of defense. Once a thread is deemed malicious, instead of blocking disk accesses, other more costly metrics can be used to improve the false positive rate without impacting performance, since it would not be computed for all other threads. As an example, future work should focus on the distribution of random (encrypted files) and constant (ransom notes) data. Once false positive rate is small enough, an interaction with the user to eliminate the last ones seem reasonable. Indeed ransomware have a very specific behavior and the average user should know if she is encrypting all its files on purpose or not. Future work should investigate which information to report to a user and if the approach is practical.

Appendix 1: Ransomware Collection

Table 3: An overview of the active ransomware families used in the experiments (i.e, 87.98%). More details at: `http://people.rennes.inria.fr/Aurelien.Palisse/DaD.html`.

Family	Samples	Family	Samples	Family	Samples
Teslacrypt	195 (24.43%)	Yakes	25 (3.13%)	Shifu	9 (1.12%)
Cerber	135 (16.91%)	Deshacop	19 (2.38%)	Fsysna	8 (1%)
Xorist	125 (15.66%)	Locky	17 (2.13%)	Shade	7 (0.87%)
Bitman	101 (12.65%)	Gpcode	13 (1.62%)	Dalexis	5 (0.79%)
Zerber	27 (3.38%)	Gamarue	9 (1.12%)	Usteal	5 (0.79%)

Appendix 2: Empirical Tests

Table 4: Shannon entropy values with 10K files for each file type.

File types	Minimum	Average	Maximum	Variance
PNG	0.14	7.87	7.99	0.33
PDF	1.45	7.74	7.99	0.16
ZIP	3.21	7.93	7.99	0.07

Table 5: Chi-Square (χ^2) values with 10K files for each file type.

File types	Minimum	Average	Maximum	Variance
PNG	275.72	1.69e+6	3.76e+9	2.74e+15
PDF	306.86	1.50e+6	5.07e+8	1.30e+14
ZIP	220.44	4.74e+5	9.11e+8	1.23e+14

References

1. Bisson, D.: C&C Servers? Too Risky! Android Botnet Goes with Twitter Instead (August 2016), https://www.bleepingcomputer.com/news/security/candc-servers-too-risky-android-botnet-goes-with-twitter-instead/
2. Bonferroni, C.E.: Teoria statistica delle classi e calcolo delle probabilita. Libreria internazionale Seeber (1936)
3. Cimpanu, C.: Microsoft Announces Controlled Folder Access to Fend Off Crypto-Ransomware. https://www.bleepingcomputer.com/news/microsoft/microsoft-announces-controlled-folder-access-to-fend-off-crypto-ransomware/ (June 2017)
4. Clonezilla: The Free and Open Source Software for Disk Imaging and Cloning, http://clonezilla.org/
5. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., Maggi, F.: ShieldFS: a Self-healing, Ransomware-aware Filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. pp. 336–347. ACM (2016)
6. Craig: Differentiate Encryption From Compression Using Math (June 2013), http://www.devttys0.com/2013/06/differentiate-encryption-from-compression-using-math/
7. Crystal Dew World: CrystalDiskMark is a disk benchmark software, http://crystalmark.info/software/CrystalDiskMark/index-e.html
8. Digital Corpora: Producing the Digital Body, http://digitalcorpora.org/
9. Geekbench: New benchmarks, redesigned interface, http://geekbench.com/
10. GNU Octave: Scientific Programming Language. https://octave.sourceforge.io/octave/function/chi2inv.html
11. Haschek, C.: How to defend your website with ZIP bombs. https://blog.haschek.at/2017/how-to-defend-your-website-with-zip-bombs.html (July 2017)
12. Ivanov, A., Sinitsyn, F.: The first cryptor to exploit Telegram (November 2016), https://securelist.com/blog/research/76558/the-first-cryptor-to-exploit-telegram/

13. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E.: UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In: Proceedings of the 25th USENIX Security Symposium, Austin Texas. pp. 757–772. Usenix (2016)
14. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 3–24. Springer (2015)
15. Kolodenker, E., Koch, W., Stringhini, G., Egele, M.: PayBreak: Defense Against Cryptographic Ransomware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 599–611. ACM (2017)
16. Malekal: Malware Repository, http://malwaredb.malekal.com/
17. Mbol, F., Robert, J.M., Sadighian, A.: An Efficient Approach to Detect TorrentLocker Ransomware in Computer Systems. In: International Conference on Cryptology and Network Security. pp. 532–541. Springer (2016)
18. Microsoft: File System Minifilter Drivers, https://msdn.microsoft.com/en-us/windows/hardware/drivers/ifs/file-system-minifilter-drivers
19. Microsoft: Windows Performance Toolkit, https://msdn.microsoft.com/en-us/windows/hardware/commercialize/test/wpt/index
20. Palisse, A., Le Bouder, H., Le Guernic, C., Legay, A., Lanet, J.L.: Ransomware and the Legacy Crypto API. In: The 11th International Conference on Risks and Security of Internet and Systems (2016)
21. PCMark 8: The Complete Benchmark for Windows 8.1, Windows 8 and Windows 7, https://www.futuremark.com/benchmarks/pcmark
22. PolarToffee: Found a sample of the AES-NI ransomware. https://twitter.com/PolarToffee (April 2017)
23. Scaife, N., Carter, H., Traynor, P., Butler, K.R.: Cryptolock (and Drop It): Stopping Ransomware Attacks on User Data. In: Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on. pp. 303–312. IEEE (2016)
24. Sebastián, M., Rivera, R., Kotzias, P., Caballero, J.: Avclass: A Tool for Massive Malware Labeling. In: International Symposium on Research in Attacks, Intrusions, and Defenses. pp. 230–253. Springer (2016)
25. SonicWall: Annual Threat Report. Tech. rep., SonicWall (2017), https://www.sonicwall.com/docs/2017-sonicwall-annual-threat-report-white-paper-24934.pdf
26. The Talos Group: MBR Filter Driver, https://github.com/vrtadmin/MBRFilter
27. Trend Micro: CryLocker Uses Imgur as C&C (September 2016), http://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-recap-sept-2-2016-crylocker-uses-imgur-as-c-c
28. Trend Micro: Cerber Starts Evading Machine Learning (March 2017), http://blog.trendmicro.com/trendlabs-security-intelligence/cerber-starts-evading-machine-learning/
29. Viper: Binary management and analysis framework., http://viper.li/
30. VirusShare: Malware Repository, https://virusshare.com/
31. Wardle, P.: Towards Generic Ransomware Detection (April 2016), https://objective-see.com/blog/blog_0x0F.html
32. Young, A., Yung, M.: Cryptovirology: Extortion-Based Security Threats and Countermeasures. In: Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on. pp. 129–140. IEEE (1996)
33. Young, A.L., Yung, M.M.: An implementation of Cryptoviral Extortion Using Microsoft's Crypto API. CiteSeerX (2005)