Ransomware Detection*

Julian Wolf
Friedrich-Alexander-University
Erlangen-Nuremberg
julian.jw.wolf@fau.de

ABSTRACT

Ransomware infections are increasing and approaches to detect ransomware and protect devices are necessary. Most approaches to detect cryptolockers rely on dynamic behavioral analysis of typical ransomware behavior like file access, filesystem activity and network activity. Some approaches work with a mix of static and dynamic analysis to detect features unique to ransomware, like some form of ransom demand. But since all of those techniques are highly specific to what is considered typical ransomware behavior it can be assumed that ransomware developers will soon adapt to detection tools and new families with different behavior will spread. After a discussion of current ransomware families, a classification of detection methods and discussion of research regarding ransomware detection, a tool evaluation is presented. Several detection tools were tested and it could be shown that with minimum effort detection could either be completely avoided or at least to a point where between 30 and 50 files could be properly encrypted before detection.

KEYWORDS

ransomware, malware, analysis, ransomware detection

1 INTRODUCTION

Ransomware, a type of malware that uses extortion to make victims pay ransom, became a very well known term after a huge increase in both variety and infected computers, especially with high-profile targets like the UK's National Health Service or the logistics company Fed-Ex [10]. A Trend Micro security report outlines an increase of 752% in new ransomware families in 2016 compared to 2015, from 29 to 247 [29], their mid-year report on 2017 reports those "unprecedented" outbreaks by WannaCry and Petya variants, and estimates as much as USD 4 billion damage done by WannaCry alone [28]. And even though this type of malware experiences this huge spike and publicity recently, the idea dates back to the 1980s, with "AIDS" (also known as PC Cyborg) being considered one of the first Ransomware reported in 1989, although the term was not yet used. Instead, in 1996 Young and Yung wrote about "Cryptovirology" and "Extortion-Based Security Threads" [31].

*This paper was written as part of the conference seminar "IT Security" which was organized by the chair of IT Security Infrastructures at the Friedrich-Alexander-University Erlangen-Nuremberg during the winter term 2018. Special thanks to Tobias Latzo for the provided support during the course of this paper.

1.1 What is Ransomware?

The malware class Ransomware in general works with extortion, but the details vary a lot depending on the platform and actual implementation. The common goal is to deny access to something important, for which the user is willing to pay ransom, therefore the terms denial of service or denial of access are also not uncommon. Depending on the device and user there are different approaches at the moment to implement ransomware, categorized in two major groups: Cryptography based ransomware and locker based ransomware [5].

Cryptographie based ransomware is the older class, and was described by Young and Yung in the following way [31]:

> A cryptovirus (cryptotrojan) is a computer virus (Trojan horse) that uses a public key generated by the author to encrypt data D that resides on the host system, in such a way that D can only be recovered by the author of the virus (assuming no fresh backup exists).

That means, the malware encrypts files in a way that only the "owner" (which does not necessary have to be the author) of the malware can decrypt them. Implementation details differ in used cryptography and the attack vector, with some malware encrypting single files and others encrypting for example the Master File Table (MFT). While earlier ransomware implementations used symmetric encryption algorithms and could be decrypted easily, modern approaches use symmetric encryption of files for performance reasons and then asymmetric encryption to encrypt the key so that the private key is held back and could be used to decrypt the symmetric key after the ransom was paid [19].

The popularity of ransomware in recent years can be explained by availability of easy to use and anonymous payment options, with Bitcoin as cryptocurrency being the most popular one recently. While the "AIDS" ransomware was distributed via floppy disk and the ransom had to be paid as cheque posted to Panama [11], modern ransomware spreads via the internet, for example via email or vulnerabilities, and the ransom is often paid to anonymous Bitcoin wallets.

The ongoing trend with an increasing number of ransomware infections per month can be seen in figure 1 as provided by the Symantec Internet Security Threat Report Ransomware 2017 [26].

With different device categories new attack vectors for ransomware open up. While traditional desktop or server systems usually hold a lot of important files and documents and are vulnerable to encryption, smartphones or embedded devices usually don't hold important files locally (or have a cloud backup), but are on the other hand vulnerable to

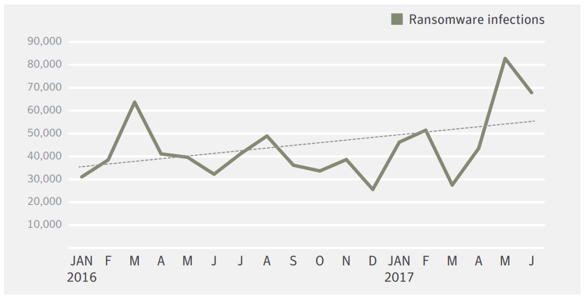


Figure 1: Ransomware infections by month, taken from [26]

attacks the completely deny access to the device by locking the user interface. This would not be an effective approach on traditional desktop or server systems since they can easily have their disk accessed through some other system, but that usually is not an easy option on mobile and embedded devices. This kind of malware is usually referred to as "Locker" or "Locker-Ransomware".

But there is also malware that pretends to be ransomware without actually being able to recover the files or devices once they are infected. While those do not technically fit into the ransomware class, they still demand the ransom and rely on users not being able to see this difference. In this paper this will be considered as part of the ransomware class since the used techniques are similar.

Another common component of ransomware, which was also already part of "AIDS", is that the message displayed to the user often uses social engineering techniques, like pretending that the user violates laws or license agreements and that the user needs to pay a small fee to avoid further fines or investigation.

Since ransomware detection relies a lot on understanding current ransomware approaches, therefore first some more current and widespread ransomware families will be discussed in the following section.

1.2 Ransomware examples

1.2.1 WannaCry. The ransomware that so far got the most publicity and attention in 2017 was called *WannaCry* [21, 27], a crypto-ransomware that cause disruption by also affecting several high-profile targets like the UK National Health Service. After infecting a system, WannaCry searches for files with specific file extensions, and encrypts them with a separate AES encryption key, and each of those keys is then encrypted with a RSA public key so that the RSA private key allows for decryption. This is equivalent to the heterogeneous approach described in the previous chapter, combining both advantages of symmetric and asymmetric encryption. WannaCry does not overwrite existing files, the new and encrypted files are created in the working directory and then renamed, adding the file extension .WINCRY, and then moved to the directory of the original file. The ransomware also tries to delete existing Windows Shadow Copies that might exist. WannaCry uses the Windows cryptography API for the RSA encryption and a statically linked third-party implementation for AES operations.

1.2.2 CryptoWall. *Cryptowall* [6] in version 3.0 has a similar approach, it uses AES for file encryption and RSA to encrypt the key. That's also the big difference to Cryptowall in version 2.0 which used public-key encryption for the files with performance disadvantages [23, 25]. For Cryptowall both public and private key are generated outside of the infected machine and the public key used for encryption is provided by a server after the infected machine opens a connection.

1.2.3 Petya and NotPetya. The *Petya* [4, 15, 24] ransomware uses a different strategy to extortion. Instead of relying on user privileges it tries to get administrative privileges via the default Windows User Access Control (UAC). So, for getting necessary privileges it relies on social engineering by asking the user to open a letter of application, as which the malware disguises itself. In it's next step Petya ransomware overwrites the Master Boot Record and makes Windows crash with an error message and forcing a reboot of the system (SeShutdownPrivilege and NtRaiseHardError). Since the MBR got modified it now startes with the Petya MBR, pretending to run a disk check while in the background encrypting the disk. After the next reboot the ransomware message is displayed to the user.

Special about Petya is that it is considered a wiper instead of ransomware because of how files are encrypted with a random key that does not have any connection to the "installation key" that is displayed to the user. So, while decryption with the right key could be possible, in reality the key can not be provided to the victim. Because of this and the fact that Petya almost exclusively targets the Ukraine Symantec considers the Petya outbreak a politically motivated attack with the goal of disruption [26].

Similar to Petya is a modified version called *NotPetya*, which has a lot of modifications. It has both the capability to just encrypt files in user mode but also the capability to modify the MBR and behave like Petya. Which approach is chosen depends on the system it is run on and which anti-virus software is running on the system [17].

1.2.4 Ordinypt. *Ordinypt* [8] is malware that pretends to do file encryption and demands ransom but in reality only overwrites data with sequences of random characters. So, in consequence, there is no way to actually recover files from this pseudo-encryption even though the ransom demand page states that it is possible and the software will be provided after payment. Therefore, Ordinypt in reality is a "wiper disguised as ransomware" but for this paper shall be considered relevant as it behaves similar to real working ransomware and the threat, complete loss of data, is basically the same. Therefore detection approaches should also consider this type of malware to prevent destructive software no matter whether the malware is actually implemented in a way that theoretically allows recovering data or not.

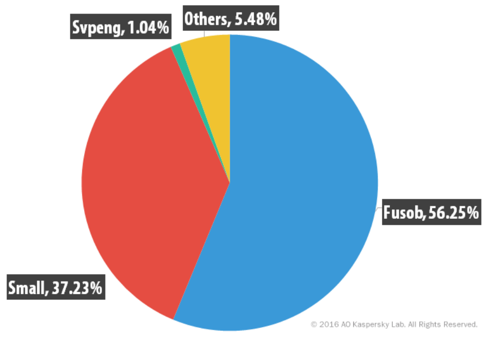


Figure 2: Mobile Ransomware 2015/2016, taken from [14]

1.2.5 Fusob and Small. The last example in this section is a locker ransomware called *Fusob*. While for desktop or server systems ransomware is in almost all cases a form of cryptolocker, because that's where important files are the target, this approach doesn't work for mobile devices that usually are backed up into the cloud or some other system, unless the attacker could also manage to destroy the backup. But for mobile devices on the other hand the ability to completely overlay the screen with a ransom message makes the device unusable to the user, and the lack of access to the device's hardware (storage) makes it hard to recover. The two dominant ransomware families for mobile devices are called Fusob and *Small* and they make up for more than 93% of mobile ransomware in 2015 and 2016 according to Kaspersky [14], as can be seen in figure 2.

Both families also rely on social engineering by pretending that the message comes from government officials or agencies like FBI or NSA.

Kaspersky's ransomware report 2014-2016 shows how much the amount of users encountering mobile ransomware has changed from between 2014 and 2016 [14], the visualization can be seen in figure 3.

1.3 Ransomware detection challenges

As the above examples have shown there's a variety in ransomware techniques, that are specific to the kind of system they target and their approach to blocking user access, so detection approaches also need to cover a wide variety of metrics. The second huge problem is that ransomware operations usually look like legitimate user operations, especially with cryptolockers which don't need special privileges and rely on cryptographic operations just like legitimate applications. Cryptolockers either implement their own cryptography or use existing libraries. Besides that, they only need to read and write files. And, as with any malware, ransomware is also an arms race between malware developers and IT security providers, where malware developers react to new countermeasures by improving their malware, which leads to new countermeasures and so on. For ransomware this means that the malware could try to behave more like legitimate software or a human user.

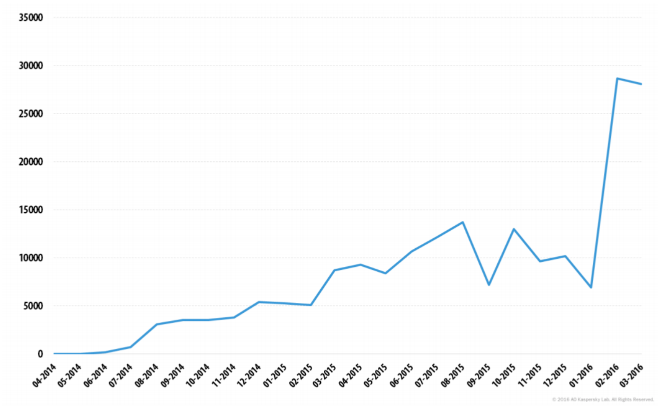


Figure 3: The number of users encountering mobile ransomware at least once in the period April 2014 to March 2016, taken from [14]

Young and Yung suggested in 1996 "auditing access to cryptographic tools" and not making cryptographic tools available to users would reduce the risk of cryptoviruses [31]:

> Incorporating strong cryptographic tools into the operating system services layer may seem like it would increase system security, but in fact, it may significantly lower the security of the system if the system is vulnerable to infection. Furthermore, with such tools readily available, virus writers would not even have to understand cryptography to create cryptoviruses.

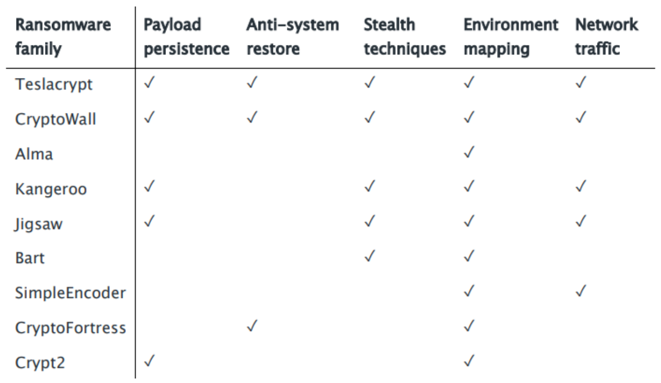
With many cryptographic libraries available and many programs, and users, relying on cryptography for security and privacy this approach is not feasible. While some cryptolockers indeed use operating system libraries, if those were not available they would ship their own or open source libraries with the malware.

But they also suggest "implementing mechanisms to detect viruses prior to or immediately following system infiltration", and this a both, a feasible and possible approach that will be covered in the following section.

2 CLASSIFICATION

Before discussing current research on ransomware detection this section will propose a classification of detection methods. In general a distinction can be made between static analysis and dynamic analysis. Static analysis is performed by looking at an executable without running it, hereby extracting information. Dynamic analysis is performed by running an executable an analyzing it's runtime behavior. Both methods have advantages and disadvantages and malware can use protection mechanisms against both, making analysis difficult.

In automated detection of malware, or ransomware in particular, static analysis plays a minor role as countermeasures



Ransomware family	Payload persistence	Anti-system restore	Stealth techniques	Environment mapping	Network traffic
Teslacrypt	✓	✓	✓	✓	✓
CryptoWall	✓	✓	✓	✓	✓
Alma				✓	
Kangeroo	✓		✓	✓	✓
Jigsaw	✓		✓	✓	✓
Bart			✓	✓	
SimpleEncoder				✓	✓
CryptoFortress		✓		✓	
Crypt2	✓			✓	

Figure 4: Ransomware behavior comparison, taken from [16]

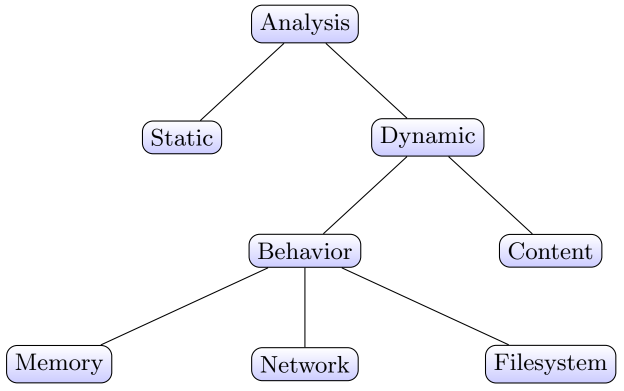


Figure 5: Ransomware detection method classification

are easy to implement. Datatabase lookups of signatures can be performed, but with self-modifying code or minor variances this method is very limited and not able to detect new strains of malware. Part of static analysis could also be to check for specific function calls, analyzing strings, or, depending on the platform, analyzing content that is shipped with the executable. On Android, for example, it is possible to look at the manifest and the permissions as part of static analysis.

Dynamic analysis on the other hand, trying to analyze runtime features of a process, has the chance of analyzing the behavior of a process and therefore the ability to also detect new or modified strains of ransomware.

Within dynamic analysis further distinction can be made between content based analysis and behavior based analysis.

For behavior based analysis there's several typical components on ransomware behavior that can be part of a detection approach: memory behavior analysis, network behavior analysis and filesystem behavior analysis. Those approaches are based on the idea that ransomware has a very specific behavior that is common to all cryptolocker ransomware implementations (typical behavior also exists for other lockers): At some point a ransom demand will be written to a file or displayed, at some point a key exchange with a server will happen, at some point files will deleted or overwritten. And even though behavior can be modified to avoid detection, in some form this behavior will be displayed by the ransomware.

Nieuwenhuizen compared different ransomware families for behavioral features [16]. The comparison focuses on very generic features that are not unique to ransomware and are, even though quite common, not inherently part of ransomware or necessary for ransomware, therefore are not part of this classification. The analysis nevertheless gives a good overview about ransomware behavior, as can be seen in in figure 4.

Dynamic analysis with the focus on content on the other hand will try to detect ransomware by analyzing file content that is read and written by a process. Most user processes have a limited number of file types used for input and a limited number of file types used for output. For example, a text editor might open text documents, maybe also images, and have text documents as output. Ransomware on the other hand has a high number of different read file types but little variety in the output. Content based analysis might also consider whether a file modification is within reasonable range, e.g. by checking similarity to the original file or by checking entropy since encrypted files show a basically equal distribution that usually cannot be seen in most other files.

The classification is displayed in figure 5.

3 DETECTION APPROACHES

The first part in this section will be about ransomware detection approaches for cryptolockers, the second part will discuss mobile (especially Android) lockers.

3.1 File encryption

Scaife et al. propose a behavioral analysis based on a typical behavioral signature of cryptographic ransomware which usually falls into one of three categories [20]:

- **Class A** ransomware reads a file, writes the encrypted data in place and closes that file again.
- **Class B** ransomware additionally moves the file out of it's directory before opening and moves it back after closing it
- **Class C** ransomware creates a new file in which the encrypted data is written, and the original file is deleted or overwritten

They use five indicators, three primary and two secondary, which combined can indicate ransomware behavior before all data is lost. The focus hereby is on the transformation of encrypted files.

The first indicator is the *filetype*. Files usually have an indicator for their filetype, a "magic number". For example PDF files start with *25 50 44 46*. If something happens that makes the filetype change this might be an indicator for ransomware, as encrypting the whole file will also change the magic numbers.

The second indicator used by the authors is *similarity*. Since cryptography aims at producing output that has no relation to the original data. Therefore, if data is changed in a way that there is little similarity, there's a good chance it got encrypted. To test for *dissimilarity* the authors use a similarity-preserving hash function, *sdhash*. Using this function they get a similarity score describing the confidence about similarity of data. A ransomware encrypted file is expected to have a similarity score of near-zero compared to the original file.

The third and last primary indicator is the Shannon *entropy*, which describes the distribution of characters in a message. While compressed or encrypted data usually has an almost perfect distribution or high entropy, other files will have a much lower entropy.

As secondary indicators Scaife et al. use *deletion* and *file type funneling*. While delete-operations per se are not suspicious, a big number of delete-operations in a user's directory, just like Class C ransomware performs it, is suspicious and potentially malicious. File type funneling means the concept of reading a big variety of filetypes but only writing a single filetype. This is not something only specific about malware since other software performs similarly (e.g. text processors), but ransomware cryptolockers strongly show this behavior. The authors consider the difference between types read and types written an indicator which can be used with a threshold for ransomware detection.

The software developed with those indicators, *CryptoDrop*, uses the union of the three primary indicators for early detection. According to the authors most malware samples tested triggered all three indicators while the majority of benign programs did so. The authors acknowledge that CryptoDrop has limitations and can't distinguish encryption or compression performed by the user from a ransomware attack, and because that leave the final decision how to proceed to the user [20].

A similar approach, called Redemption, is suggested by Kharraz and Kirda [12]. They suggest two categories of features used for detection: Content-based and behavior-based. In the content-based group they collect the features entropy ratio of data blocks, file content overwrite, and delete operation. In the behavioral category they monitor directory traversal, converting to a specific filetype, and access frequency. From these indicators the malice score of a process is calculated.

The big difference though is that the authors don't just rely on early detection of ransomware for minimal damage, but also try to avoid any damage by completely preventing file encryption/deletion. Therefore, Redemption uses a protected memory area where all write requests on user files are redirected. If the malice score of a process exceeds the threshold the user is notified and the real user files are only overwritten if the user confirms that this is not due to the action of malware.

Additionally, Kharraz et al. suggest to use decoy files to detect malicious file access. Those decoy files would be spread all over the filesystem and if accessed could indicate ransomware [13]. This is also proposed as a method for intrusion detection in general by Yuill et al. where those files are called *honeyfiles* [32]. The difficulty here for ransomware detection is that decoys are only useful if they are hit by the ransomware very early, and that file access patterns depend a lot on the implementation of the malware. Any patterns, also depending on accessed file types, from bottom up, top down, new to old, small to big, and vice versa, as well as many others, are possible and would have to be covered by decoy files. While this is almost impossible to completely cover, they would still bring the advantage of detection even though maybe not at the earliest possible point.

3.1.1 Network based. A very different method, based on network connections, is discussed by Ahmadian et al. [1]. The authors first take a look at ransomware from a point of view how perfect extortion looks like and provide a taxonomy. *Non-Cryptographic Ransomware (NCR)* is considered a weak technique as there is countermeasures to reverse the damages caused. As a second group they identify *Cryptographic ransomware CGR* with the categories *Private-key cryptosystem ransomware PrCR*, which is based on symmetric cryptography and can easily be breached by analysts, and *Public-key cryptosystem ransomware PuCR*, where asymmetric encryption with a public and a private key is used. The authors conclude that this kind of ransomware is flawed due to the fact that the public key is in the malware's payload and a single system can't be freed without giving the private key away. This could be improved by having several key pairs, but without having a key pair per infection this method is never a good option. Additionally asymmetric encryption at the target system is generally slower than symmetric encryption. At last they consider *Hybrid cryptosystem ransomware (HCR)* where files are encrypted with a unique symmetric key per system, which then gets encrypted with a public key. The system or files can be release by sending the encrypted key to the malware writer who decrypts it using their private key and then returns the key to decrypt the files or system.

Ahmadian et al. also define a term *High survivable ransomware (HSR)* by requirements that need to be met to have "effective mass extortion means":

- Infected computers should be considered compromised and harmful
- The ransomware author should be the only person to reverse the infection, therefore anything that gives the victim a chance to reverse the infection, like generating a private key on the infected system
- Freeing one victim should not help other victims

This leads to the final definition of HSR:

> A ransomware has the "high survivability" property if it can maintain control over a critical host resource RC such that it grants access to RC solely when it is needed, and

such that if ransomware is modified or removed, RC is rendered permanently inaccessible and the decryption process can be completed only by the Command & Control server (C&C) key while the ransom is paid.

All current HSR can be found in the HCR category, therefore they all need to query a server for a unique public key to encrypt the symmetric encryption key. To do this malware often uses so called *Domain Generation Algorithm (DGA)*, where a domain name (with a C&C) is dynamically generated through an algorithm. This is used to make it harder to take down a network behind a malware attack, since it requires significantly more effort, and is more resilient, than a hardcoded list of C&C domains.

The authors suggest a connection monitor that checks for DGA DNS requests based on character transitions and recognizing a gibberish query which is then blocked (Connection Monitor Verifier CMV). A blocked connection for ransomware then means that the files won't be encrypted or at least that the key itself can't be encrypted. Additionally, signing domains for valid connection requests is proposed.

According to Ahmadian et al. their framework was able to discover all current ransomware families in the HSR category.

A similar approach is taken by Cabaj and Mazurczyk [7]. The authors in this case use *Software-defined networking (SDN)* to check the DNS traffic and compare requests to a database of known ransomware proxy servers, either by sending a copy of DNS packages to a SDN controller or by routing all DNS packages through a SDN controller. Blocking or recording the malicious ransomware traffic then provides the encryption key in case of symmetric encryption or can block the ransomware from receiving the needed public key.

Those two approaches show that analyzing network behavior is possible to detect malware in general, and also ransomware. Yet the assumptions made are limiting the effectivity of this method. First, non-cyptographic ransomware might be weak on desktop and server systems, but as will be discussed in section 3.2, is not that weak on mobile or embedded systems where the memory can't be accessed easily. Second, the approach of only considering "perfect" ransomware from the HSR category a real threat completely ignores malware that does not fit into this category, like wipers. Ordinypt, as discussed earlier, is a wiper pretending to be ransomware, does not need network communication and therefore would not be detected. Another aspect is that malware could be implemented to act like a wiper if network communication is blocked or disturbed, which would result in file loss.

Therefore, it is clear network communication behavior is something that works as an indicator for malware, but should not be considered a standalone solution for ransomware detection.

3.1.2 Dynamic analysis. Sgandurra et al. suggest a broader dynamic analysis that covers several features like API invocations, registry keys, file and directory operations, dropped files, and embedded strings [22]. Their detection approach, EldeRan, is based on the idea that ransomware in general will always behave in a very similar way, therefore they used machine learning to extract those relevant features. While they initially had over 30000 features, the most relevant ones were quite distinct as can be seen in table 1.

Features	Top 400	Top 100
Registry Keys Operations	48.25%	49%
API Stats	24.00%	27%
Strings	8.25%	5%
File Extensions	8.00%	9%
Files Operations	5.25%	6%
Directory Operations	4.00%	2%
Dropped Files Extensions	2.25%	2%

Table 1: Percentage of the most relevant features for each class, taken from [22]

3.2 Android and Embedded

Different environments face different challenges in regards to malware as well, therefore also with ransomware. As mentioned earlier, in the case of mobile or embedded devices the locked resources are generally not files or documents but devices that are hard or costly to repair or replace. This is not only the case for smartphones. With the concept Internet of Things (IoT) spreading and more and more devices, like fridges or ovens, being connected to networks/the internet, therefore being vulnerable to malware and ransomware, those devices definitely could be held to ransom as well. On Android static analysis is different because of different programming languages that are used (native code, Java) and control flows between individual components. Dynamic analysis is different in ways that it is in general event triggered where external events, like SMS or sensor data, user interaction and the system environment play a big role for malware to hide itself, using timing features and environment analysis [18]. Additionally, Android uses a granular permission system for apps in which the permissions are declared in the manifest file. The *ScarePakage* ransomware uses permissions to lock the device, kill other tasks, keep the device from going to sleep and to use the internet to verify payment.

Yang et al. use four factors for static analysis of an app, as taken from the extracted apk file:

- Permission
- Sequence of API invocations
- Resources
- APK structure

But since static analysis can not deal well with obfuscation and encryption, the authors additionally suggest dynamic analysis and add another four features for analysis during runtime [30]:

- Critical path and data flow
- Malicious domain access
- Malicious charges (e.g. via SMS or calls)
- Bypassing Android Permission

This combined approach of dynamic and static analysis is also used by Andronio et al. for the detection tool HELDROID [2].

The authors designed HELDROID with three main components: The *Threatening Text Detector*, the *Encryption Detector* and the *Locking Detector*. The constraint is that both scareware and ransomware would use threatening text, so if a sample does not trigger the *Threatening Text Detector* it is not considered ransomware or scareware. The performed text analysis is done both static and dynamic, first from inspecting the binary and resource files and then running the app in a sandbox while inspecting allocated memory and communications. The analysis for encryption is done through static flow analysis and corresponding API calls. The *Locking detector* works with heuristics on execution paths and is performed static as well.

To counter these detection methods malware could deliver the threatening text via images and make those images hard to decipher for text detection algorithms (like a captcha) and deliver it's own encryption functions instead of the Android API. But the authors claim no such malware has been found to they point where they conducted their research.

Azmoodeh et al. suggest a different method for detecting cryptlocker ransomware on IoT devices: Based on a processes energy footprint. With this approach the authors only target cryptolockers since those have a unique energy fingerprint, and lockers on the other hand wouldn't have this. The authors used machine learning of existing cryptolocker samples on different Android devices to generate ransomware fingerprints and showed that it is possible to distinguish between goodware and cryptolockers based on this criteria [3].

4 TEST OF AVAILABLE DETECTION TOOLS

After handling the theoretical background of ransomware this section will point out some tests that were done with existing available ransomware detection tools. Even though there is suggestions about detecting ransomware with network monitoring no evidence could be found that this is used in any of the available tools. Additionally, even malware that acts more like a wiper than ransomware (e.g. Ordinypt) should be detectable even though it does not use a key exchange.

A simple ransomware (*Jamsomware*) was implemented so that signature based detection would not be possible and detection methods could be tested. The focus hereby was not to develop good or the best ransomware but more to develop a simple implementation that uses the most common techniques. Jamsomware falls in the Class C according to Scaife et al., it creates a new file and then deletes the original file. It is fully implemented in Python (packed as a single executable file with *pyinstaller*) using the Cryptodome library for AES symmetric encryption of files with a key randomly generated on the system.

Instead of encrypting the symmetric key with an asymmetric encryption algorithm it is just stored in plain text on the disk. This is based on the assumption that encrypting the symmetric key again would not bring any benefit for detection. Therefore, there also is no key exchange with a server (for either receiving a public or sending a private key) but with the focus on local filesystem based detection this neither influences the experiment.

It takes several parameters that could be a directory as entry point or an existing key. If it is started without any parameters it will create a new random key and start iterating through the current user's home directory. Encountering a directory it creates a copy of the original binary and starts it as a new process with the encountered directory and the encryption key as parameters, found files matching a pattern for the name extension get encrypted immediately.

All created subprocesses will delay their execution randomly for 5-10 seconds and every subprocess behaves just like the original one, encrypting files and creating new subprocesses (with their own executable each) for every directory. Every subprocess also deletes its executable after finishing.

The goal of this multiprocessing with separate executables implementation is to disguise the malicious behavior and to not raise thresholds for single processes. Additionally, this recursive multiprocessing approach (with random delays) has the advantage that filesystem traversal is less obvious and that a single blocked process or quarantined executable will only prevent Jamsomware from accessing a branch of the filesystem tree, not completely stop it. This could be further improved in the future by first creating all new recursive processes before attacking files.

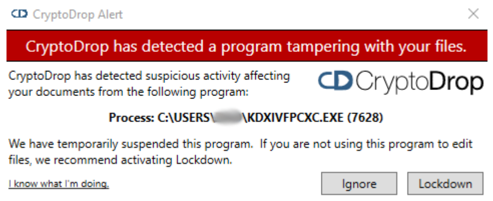
Another technique Jamsomware uses to disguise its operations is not encrypting the first block (16 Bytes) of every file. This is based on the CryptoDrop indicator that checks for changed filetypes.

The target machine for testing was a virtual Windows 10 system with a relatively small user home directory of 296 relevant files from different types targeted by *Jamsomware*, (text-) documents and media files. The files are spread in 30 (sub-) directories with directories containing between one and over 100 files.

Four different protection tools were tested with Jamsomware: *CryptoDrop, McAfee Ransomware Interceptor, Avast Premier Behavior Shield, Bitdefender Anti-Ransomware Tool.* All tools are commercial closed source applications and with the exception CryptoDrop (see section 3.1) no knowledge about the detection approach of those tools was available.

CryptoDrop[1] discovered the ransomware after some minutes (depending on the file structure and random delays), suspended the process and asked for user confirmation to enter lockdown mode where the protected area is mapped read-only and files therefore can't be deleted anymore. A lockdown mode has the advantage that no matter how many malicious processes are running and disguised, one being discovered is enough to protect the files. Requiring user input before entering this mode has the disadvantage of delayed reaction to ransomware discoveries, but a compromise in regards to user acceptance where lockdown mode in case of a

[1]https://www.cryptodrop.org/


CryptoDrop Alert

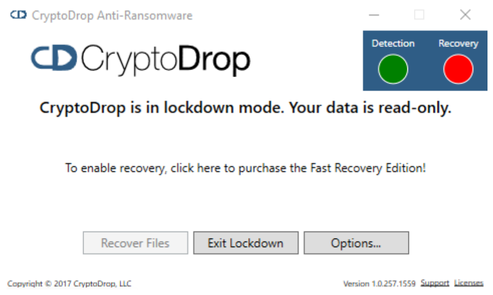
CryptoDrop has detected a program tampering with your files.

CryptoDrop has detected suspicious activity affecting your documents from the following program:

Process: C:\USERS\[redacted]\KDXIVFPCXC.EXE (7628)

We have temporarily suspended this program. If you are not using this program to edit files, we recommend activating Lockdown.

Figure 6: Cryptodrop suspended process


CryptoDrop Anti-Ransomware

CryptoDrop is in lockdown mode. Your data is read-only.

To enable recovery, click here to purchase the Fast Recovery Edition!

Copyright © 2017 CryptoDrop, LLC

Version 1.0.257.1559

Figure 7: Cryptodrop in lockdown mode

false positive would not find acceptance. Cryptodrop's reaction to discovering ransomware can be seen in figures 6 and 7. As can be seen there Cryptodrop also offers a restoration option to recover lost files, but this a premium feature not available in the free version and was not further investigated as part of this work.

In total between 36 and 44 files could be encrypted and deleted before this happened. Several files only got encrypted but the delete operation on the original file couldn't be performed anymore. Another approach which didn't use a delete operation but instead tried to overwrite the original files with random characters with a distribution similar to the one of the character frequency of the English language to influence the entropy. In this case only two files could be overwritten before CryptoDrop recognized the file tampering.

The McAfee Ransomware Interceptor pilot version[2], which "is an early detection tool that tries to prevent file encryption attempts by ransomware malware", was not able to discover *Jamsomware*. While there is little information on how it works it claims not being "a static detection tool" but indeed being for early detection and prevention, therefore it was expected to detect the file encryption. Similar results could be observed with Bitdefender Anti-Ransomware Tool[3] where all files could be encrypted and deleted without detection.

Avast Premier, which includes a module for ransomware detection, Behavior shield[4], detected Jamsomware as well with around 50 encrypted and deleted files.

The results can also be seen in figure 8.

[2]https://www.mcafee.com/us/downloads/free-tools/how-to-use-interceptor.aspx
[3]https://www.bitdefender.com/solutions/anti-ransomware-tool.html
[4]https://blog.avast.com/behavior-shield-our-newest-behavioral-analysis-technology

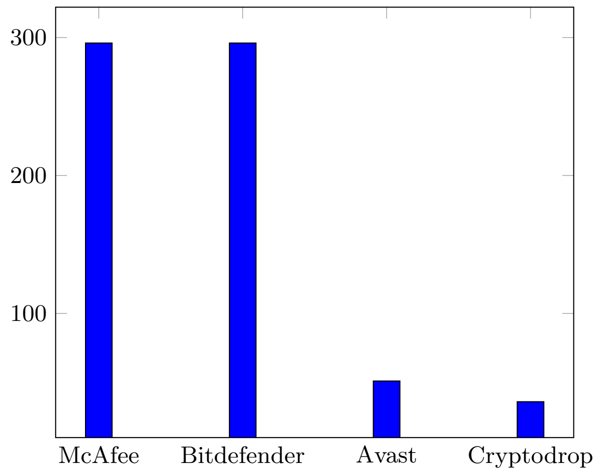


Figure 8: Number of encrypted and deleted files out of a total of 296

It could be shown that with very little effort ransomware detection could be partially avoided or delayed for long enough to have several files encrypted and that relying on those tools alone is not sufficient to protect important data from malware.

Several other features to prevent detection could be additionally implemented but are out of scope for this paper. While the multiprocessing approach seems very promising there's a lot of room for improvement. Future work might include an approach using a single process per file, in a coordinated matter, or implementing smarter methods of deleting the original content after encryption.

5 CONCLUSION

Different research has be done on the topic of ransomware detection and already covered a variety of methods on how to discover malware from the ransomware family, both lockers and cryptolockers, on different platforms. For Android systems where lockers play a bigger role than cryptolockers a mix of static and dynamic application analysis on several factors like strings, application permissions, control flow and API calls.

For desktop and server systems, where cryptolockers are a dangerous threat the detection approaches mostly work with behavioral analysis by using behavioral patterns of current and past ransomware to identify such processes from mere process behavior.

The downside in both cases is that all analyzed approaches are very specifically tailored to how ransomware works today. In most cases common techniques of ransomware were used to build a detection system tailored to those criteria.

Trying to catch the key exchange via network communication can be a successful approach if that is what the malware does, if the traffic can be identified and intercepted, and if the malware doesn't have a module for the case of key interception - like encrypting with a random key that would

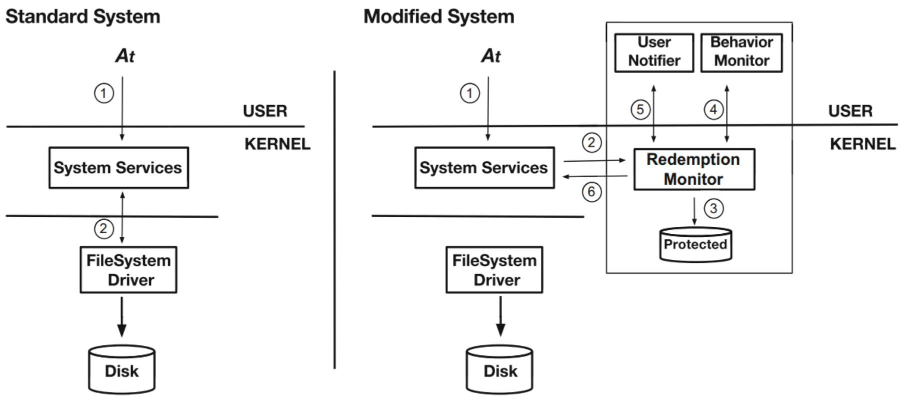


Figure 9: Redemption protective model, taken from [12]

lead to data loss. This is also not something that would help with a wiper pretending to be ransomware, like Ordinypt, and therefore would cause complete data loss as well.

Additionally, most behavioral detection approaches try to detect ransomware as soon as possible - but can't really prevent loss up to the point where the malware is detected and stopped. This could be shown using Jamsomware to test available ransomware detection tools with the result that two out of four tools didn't detect the ransomware at all and the other tools detected the ransomware too late to prevent data loss.

Therefore, additional safety barriers to protect user data from encryption should be used as shown by redemption where potentially bad actions are executed in a different memory region and only transferred if not discovered as coming from a malicious process. This has been shown as part of Redemption as discussed in section 3.1 and also been proposed by Continella et al. as ShieldFS, "A Self-healing, Ransomware-aware Filesystem" [9]. Just like Redemption, ShieldFS shadows write operations while at the same time trying to identify malicious processes.

In both cases a module is introduced protecting files from malicious processes by introducing a protective layer and having processes work on copies while still keeping a original file in a protected region. The implementation used by Redemption can be seen in figure 9. Instead of passing write requests to the original files, those requests are performed in the protected area and monitored for potential ransomware behavior. Operations that are not considered harmful, like creating a new files, are not changed and performed like before.

But even then those systems rely on detection before writing the data through to the original file. With detection tools catching up on ransomware, measures to avoid detection will be implemented and the arms race continues. And since all discussed methods are just reactive to how ransomware works at the moment, new families will emerge that won't be detectable due to different behavior and cause damage unless new protective systems for user data or devices will be implemented.

Until better protection techniques are developed a combined strategy of static and dynamic detection methods, with as many behavioral factors as possible, should become part of basic protective procedures. But real protection, where no data loss through ransomware happens, requires that data is properly backed up to locations that are not accessible for processes running with user privileges.

Additionally, kernel level security is just as relevant as detecting ransomware running as a user process. If ransomware manages to run with kernel privileges, either by tricking a user or exploiting a vulnerability, all protective models and detection approaches can be disabled and data loss cannot be prevented. In this case, offline backups are what can prevent a catastrophic outcome for users and companies.

REFERENCES

[1] M. M. Ahmadian, H. R. Shahriari, and S. M. Ghaffarian. 2015. Connection-monitor connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. In *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. 79–84. DOI:http://dx.doi.org/10.1109/ISCISC.2015.7387902

[2] Nicoló Andronio, Stefano Zanero, and Federico Maggi. 2015. Heldroid: Dissecting and detecting mobile ransomware. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 382–404.

[3] Amin Azmoodeh, Ali Dehghantanha, Mauro Conti, and Kim-Kwang Raymond Choo. 2017. Detecting crypto-ransomware in IoT networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing* (23 Aug 2017). DOI:http://dx.doi.org/10.1007/s12652-017-0558-5

[4] Tim Berghoff. 2016. Ransomware Petya - a technical review. (2016). https://www.gdatasoftware.com/blog/2016/03/28226-ransomware-petya-a-technical-review Accessed: 06.12.2017.

[5] Akashdeep Bhardwaj, Vinay Avasthi, Hanumat Sastry, and GVB Subrahmanyam. 2016. Ransomware digital extortion: a rising new age threat. *Indian Journal of Science and Technology* 9 (2016), 14.

[6] Krzysztof Cabaj, Piotr Gawkowski, Konrad Grochowski, and Dawid Osojca. 2015. Network activity analysis of CryptoWall ransomware. *Przeglad Elektrotechniczny* 91, 11 (2015), 201–204.

[7] K. Cabaj and W. Mazurczyk. 2016. Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall. *IEEE Network* 30, 6 (November 2016), 14–20. DOI:http://dx.doi.org/10.1109/MNET.2016.1600110NM

[8] Catalin Cimpanu. 2017. Ordinypt Ransomware Intentionally Destroys Files, Currently Targeting Germany. (2017). https://www.bleepingcomputer.com/news/security/ordinypt-ransomware-intentionally-destroys-files-currently-targeting-germany/ Accessed: 06.12.2017.

[9] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. 2016. ShieldFS: A Self-healing, Ransomware-aware Filesystem. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications (ACSAC '16)*. ACM, New York, NY, USA, 336–347. DOI:http://dx.doi.org/10.1145/2991079.2991110

[10] European Union Agency for Law Enforcement Cooperation (Europol). 2017. INTERNET ORGANISED CRIME THREAT ASSESSMENT (IOCTA) 2017. (2017). https://www.europol.europa.eu/activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2017 Accessed: 06.12.2017.

[11] Nikolai Hampton and Zubair A Baig. 2015. Ransomware: Emergence of the cyber-extortion menace. (2015).

[12] Amin Kharraz and Engin Kirda. 2017. Redemption: Real-Time Protection Against Ransomware at End-Hosts. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 98–119.

[13] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. 2015. *Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks*. Springer International Publishing, Cham, 3–24. DOI:http://dx.doi.org/10.1007/978-3-319-20550-2_1

[14] Kaspersky Lab. 2016. KSN Report: Mobile ransomware in 2014-2016. (2016). https://securelist.com/files/2016/06/KSN_Report_Ransomware_2014-2016_final_ENG.pdf Accessed: 06.12.2017.

[15] Malwarebytes Labs. 2017. Petya - Taking Ransomware To The Low Level . (2017). https://blog.malwarebytes.com/threat-analysis/2016/04/petya-ransomware/ Accessed: 06.12.2017.

[16] Daniel Nieuwenhuizen. 2017. A behavioural-based approach to ransomware detection. *Whitepaper. MWR Labs Whitepaper* (2017).

[17] Erin Noerenberg. 2017. NotPetya Technical Analysis. (2017). https://logrhythm.com/blog/notpetya-technical-analysis/ Accessed: 06.12.2017.

[18] Siegfried Rasthofer, Irfan Asrar, Stephan Huber, and Eric Bodden. 2015. How current android malware seeks to evade automated code analysis. In *IFIP International Conference on Information Security Theory and Practice*. Springer, 187–202.

[19] Kevin Savage, Peter Coogan, and Hon Lau. 2015. The evolution of ransomware. *Symantec, Mountain View* (2015).

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

[21] Secureworks. 2017. WCry Ransomware Analysis. (2017). https://www.secureworks.com/research/wcry-ransomware-analysis Accessed: 06.12.2017.

[22] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C. Lupu. 2016. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *CoRR* abs/1609.03020 (2016). arXiv:1609.03020 http://arxiv.org/abs/1609.03020

[23] Sophos. 2015. The current state of ransomware: CryptoWall. (2015). https://news.sophos.com/en-us/2015/12/17/the-current-state-of-ransomware-cryptowall/ Accessed: 06.12.2017.

[24] Jason Sumalapao. 2016. PETYA Crypto-ransomware Overwrites MBR to Lock Users Out of Their Computers. (2016). http://blog.trendmicro.com/trendlabs-security-intelligence/petya-crypto-ransomware-overwrites-mbr-lock-users-computers/ Accessed: 06.12.2017.

[25] Symantec. 2016. Ransom.Cryptowall. (2016). https://www.symantec.com/security_response/writeup.jsp?docid=2014-061923-2824-99&tabid=2 Accessed: 06.12.2017.

[26] Symantec. 2017. Internet Security Threat Report ISTR Ransomware 2017. (2017). https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-ransomware-2017-en.pdf Accessed: 06.12.2017.

[27] Symantec. 2017. Ransom.Wannacry. (2017). https://www.symantec.com/security_response/writeup.jsp?docid=2017-051310-3522-99&tabid=2 Accessed: 06.12.2017.

[28] Trend Micro. 2017. 2017 Midyear Security Roundup: The Cost of Compromise. (2017). https://documents.trendmicro.com/assets/rpt/rpt-2017-Midyear-Security-Roundup-The-Cost-of-Compromise.pdf Accessed: 06.12.2017.

[29] Trend Micro. 2017. TrendLabs 2016 Security Roundup: A Record Year for Enterprise Threats. (2017). https://documents.trendmicro.com/assets/rpt/rpt-2016-annual-security-roundup-a-record-year-for-enterprise-threats.pdf Accessed: 06.12.2017.

[30] Tianda Yang, Yu Yang, Kai Qian, Dan Chia-Tien Lo, Ying Qian, and Lixin Tao. 2015. Automated detection and analysis for android ransomware. In *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*. IEEE, 1338–1343.

[31] A. Young and Moti Yung. 1996. Cryptovirology: extortion-based security threats and countermeasures. In *Proceedings 1996 IEEE Symposium on Security and Privacy*. 129–140. DOI:http://dx.doi.org/10.1109/SECPRI.1996.502676

[32] J. Yuill, M. Zappe, D. Denning, and F. Feer. 2004. Honeyfiles: deceptive files for intrusion detection. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004*. 116–122. DOI:http://dx.doi.org/10.1109/IAW.2004.1437806