MWR Labs Whitepaper

A behavioural-based approach to ransomware detection

Daniel Nieuwenhuizen

MWR LABS

Contents page

1. Introduction

The rise of ransomware as a cybersecurity threat is nothing short of spectacular – from its dormant introduction nearly three decades ago, to present day, where ransomware is widespread and has become a serious threat.

Ransomware is a type of malicious software (malware) that once executed on a computer system, hinders the user from using the computer or its data, demanding a sum of money (ransom) for the restoration of the computer. Currently, ransomware attacks hinder computer operation in three ways: by blocking accessing to the computer, this form of ransomware is referred to as locker ransomware; by making user data unusable by means of employing encryption algorithms, referred to as crypto ransomware; and a combination of locker/crypto ransomware where a user is blocked from using their computer while their data is being encrypted.

Between the locker and crypto types, crypto ransomware has proven to be the most destructive. Crypto ransomware typically uses strong encryption algorithms, meaning it is most often impossible to restore (decrypt) the system without paying the ransom. This is in contrast to locker ransomware in which functionality can generally be restored with some technical knowhow[1]. As such, locker ransomware has fallen out of favour, with crypto ransomware monopolising and propelling the ransomware industry forward.

The development, distribution and successful attacks of ransomware has grown exponentially over the past 4 years. According to Trend Micro research [1], 2016 saw a record 400% rise in new ransomware families (roughly 150 new families). It is thus evident that major market share anti-virus solutions are failing to contain the threat of ransomware. The inadequacies of current solutions lie in their heavy reliance on static-based detection techniques.

MWR InfoSecurity has been performing deep research into hundreds of different ransomware families in pursuit of a way to appropriately defend against it. The techniques described in this paper are used in **RansomFlare**, MWR's state-of-the-art ransomware prevention agent that utilises dynamic (behavioural) analyses and machine-learning techniques.

This paper provides motivation for the use of machine-learned behaviour for ransomware detection, and in doing so demonstrates the technical underpinnings of RansomFlare.

[1] In most cases a safe-mode boot can restore functionality, which gives the means to remove the ransomware.

2. The strengths of ransomware

Before considering how to defend against ransomware, one must understand the problem of ransomware. The root of ransomware's success lies in its highly profitable business model. Roughly 3% [2] of US companies paid the ransom sum. This may seem low but still makes ransomware an effective way of extorting money.

Besides its high return on investment, ransomware is an appealing avenue for cyber attackers as it is seemingly immune to high-reward trade-offs as typically found in other malicious (or legitimate) business practices. High-reward typically implies high risk and/or high effort, however ransomware attackers have little chance of being caught, and once the ransomware is developed, extorting money is relatively easy.

On a technical level, the recent success of ransomware can be attributed to three main proponents:

+ **Anonymous and seamless payment system** – Crypto-currencies offer the ability to quickly move money anywhere in the world with decent anonymity if used correctly. This is highly beneficial for ransomware attackers, which can operate on a global scale while remaining financially untraceable.

  When crypto-currencies as a form of ransom payment were initially used roughly four years ago, it obtained only marginal success. Victims were unwilling to pay a crypto-currency ransom sum as public opinion did not see crypto-currencies as a legitimate financial system. However, with the rise of crypto-currency growth and exposure, came about a positive shift in public sentiments

  This shift in public sentiments have cemented crypto-currencies as the dominant form of ransom payment, which greatly benefits ransomware attackers.

+ **Low effort development** – New ransomware families and variants are discovered on a daily basis. This is partly because ransomware is relatively easy to develop. With the availability of standard cryptographic libraries, coding encryption schemes that use RSA and AES is a trivial task. Furthermore, even poorly developed ransomware (with weak encryption routines), have been found to be effective at extorting money purely by means of using scare tactics[2].

  With the growing trend of Ransomware-as-a-Service (RaaS), even non-technical attackers are able to quickly generate customisable ransomware. With RaaS, ransomware developers make an easy-to-use ransomware development kit available, which clients can buy and use to create ransomware that pays out to their own crypto-currency address.

[2] Scare tactics include threatening to delete files or delete the decryption key after a specified time has lapsed.

Further aiding in the development of ransomware are open-source ransomware projects such as EDA2[3] and Hidden Tear[4]. Initially intended for educational purposes, open-source ransomware has been used as a template for the generation of hundreds of different ransomware variants.

+ **Cost effective distribution** – Ransomware attackers make use of paid malicious distribution services to effortlessly distribute ransomware on a global scale. These distribution services use a variety of platforms such as spam, drive-by-downloads, malvertising and exploit kits.

  On a successful ransomware payload drop, no additional work is required by the attacker to extort money. The authentication of a ransom payment and issuing of the decryption key (if any) is typically automated using automated email responses. This is in stark contrast to targeted attacks where attackers have to exfiltrate data and understand the value of the data that was stolen, or understand the particular organisation and work out ways to extract money from them.

The continued success of a cyber-security threat hints at a weakness in the defences of the average computer system. Under the NIST cyber security framework [2], five key tactical core facets are required for threat mitigation: identify, protect, detect, respond and recover. This paper delves into the specifics of the *detection* of ransomware. To begin, we start off with a description of detection mechanisms found in typical commercial virus scanners.

[3] https://github.com/utkusen/eda2

[4] https://github.com/goliate/hidden-tear

3. Static-based analysis detection

Detection of malware using static-based analysis means analysing an application's code prior to its execution to determine if it is capable of any malicious activities. If the static analysis finds any malicious code, the executable will be stopped from launching.

The most common type of static analysis, which is commonly used in commercial virus scanners, is referred to as signature analysis. In signature analysis, code string patterns (signatures) are extracted from the target application's code and compared to a repository of known malicious code patterns.

Signature-based detection relies on an enormous repository of malicious code signatures. This repository needs to be frequently updated to remain current, which is by no means a trivial task. Commercial virus scanners typically have large teams of cybersecurity researchers that continuously discover, investigate and extract malicious signatures.

3.1 Limitations of static analysis

The fundamental flaw of signature-based detection is its inability to detect unknown malware which has yet to be turned into a signature. A malicious executable is only detectable once it has first been reported as malicious and added to the malicious signature repository. The repercussion of this is three fold for static-based detection efficacy:

+ **Ineffective against code obfuscation** – To bypass signature-based detection, malware developers utilise code obfuscation techniques to iteratively modify the malware so each version appears different. Critically, this code obfuscation does not affect the intended malicious behaviour, it only affects how it is statically perceived by a human or signature-based detection system.

  For ransomware attacks, code obfuscation techniques are typically used when generating the payload on the server side in the form of a "malware factory". A malware factory is the method of automatically generating large volumes of unique-hash malware variants of the original malware code using code obfuscation. An example of a malware factory attack was seen in the Cerber ransomware [4], where the attack server was able to generate new unique looking samples every 15 seconds.

  The other form of code obfuscation occurs on the victim's side where the malware unpacks a unique variant of its payload executable every time it executes. This type of malware is referred to as self-morphing as it is able to replicate morphed versions of itself. Ordered by level of complexity to develop, self-morphing malware falls into three categories: oligomorphic, polymorphic and metamorphic. Refer to [5] for further technical details.

  Currently, self-morphing obfuscation is scarcely found in ransomware. However, as development continues on ransomware, one can expect an increase in self-morphing ransomware. For a good

example of self-morphing ransomware, refer to the Virlock family [6], which evades static detection using polymorphism.

+ **Ineffective against high variant output** – Signature-based analysis is less effective on malware types that have rapid development cycles and variant output. This is problematic for signature-based detection systems as new looking ransomware is being developed far faster than new signatures can be created, tested and added to the malicious signature repositories.

+ **Ineffective against targeted attacks** – This is particularly concerning in the case of ransomware, where targeted ransomware attacks are occurring more frequently. Instead of distributing newly developed ransomware, developers may choose to first unleash new ransomware on very specific, well selected organisations. Refer to [3] for an example of a targeted ransomware attack.

It is thus apparent that for the successful containment of ransomware, a paradigm shift in the way it is detected must occur. Detecting new ransomware should be seen as not only feasible, but a critical requirement. To this end, we introduce dynamic-based analysis detection, a modern approach that aims to judge an executable not by its appearance, but by its actions.

4. Dynamic-based (behavioural) analysis detection

Dynamic-based analysis detection entails the live monitoring of processes, in order to determine if any are behaving with any malicious intent. Any maliciously behaving process will be flagged as dangerous and terminated.

4.1 Static analysis vs Dynamic analysis

One can generalise the difference between static and behavioural analysis for detection in the following manner: in static analysis, inference of the behavioural traits are made from the binary file of an unknown executable. This inferred behaviour is then used by a simple matching algorithm to assign a threat level (i.e. safe or malicious). In behavioural analysis, the behavioural characteristics of the executable is known as it is being observed in real-time, and inferences is made by an inductive decision algorithm on the threat level.

The key difference between static and behaviour based detection is the point at which inference is made – static analysis infers behavioural traits from the observed binary file, dynamic behaviour infers a threat level from observed behaviour.

This is an important difference as static obfuscation techniques alter how behaviour is inferred without actually affecting the true behaviour, thus rendering it redundant against behavioural-based detection.

In behavioural-based detection, all executables are treated as unknown, where it is up to the executable to prove it is acting in a safe, non-malicious manner. In doing so, the ability of detecting zero-day (unknown) attacks are substantially improved.

4.2 Ransomware behaviour

The fact that ransomware can be easily defined and sub-categorised within malware, implies there exists a well-defined behavioural construct in which we can predict an unknown process is ransomware.

Behavioural-based analysis has been found to be highly effective for crypto ransomware detection because it exhibits core behavioural traits necessary for a data encryption attack that does not change from variant to variant or family to family. These behaviour traits can be categorised into two distinct tasks, the suspicious setup procedure and data encryption.

4.2.1 Suspicious setup behaviour

Ransomware shares many behavioural traits with other malware, particularly in the way it installs itself before delivering the payload.

This shared behaviour can be viewed as a general “recipe-for-success” that is followed by malware developers, which can be categorised into six behaviour traits:

- **Payload persistence –** To ensure an attack is carried out to completion, it needs to persist across reboots and be able to resume upon starting. Common techniques used by ransomware includes placing a copy of its executable into the Windows startup directory, adding a registry run key entry or setting up a scheduled task, to name a few.

- **Anti-system restore** – To ensure that any malicious actions cannot be undone, malware may try to disable system restore functionality. Ransomware for example, has been known to delete Windows shadow copies, which prevents encrypted data from being restored to an older unencrypted version.

- **Stealth techniques** – malware will try to execute in a stealthy manner to avoid being noticed by the user or detected by virus scanners. Common techniques include: injection into legitimate processes, executing from the %AppData% directory and using executables named the same as common Windows executables, to name a few.

  Environment mapping – When malware is executed, it may map its system environment before initiating its setup procedure. This is typically done to determine if it's running on a real computer or on a sandbox environment that could be attempting to analyse it.

  Environment mapping is also used to determine security settings/policies, geographic location, user language, file system architecture and network drives. Certain decisions about whether to continue executing may rely on any of the environmental checks performed.

- **Network traffic** – Ransomware that requires an internet connection, does so for two possible tasks: downloading of payload related files, and/or for the communication of the encryption key.

  To ensure malicious command and control servers do not easily get shut down by authorities, malware developers use certain domain name registration tactics. Tactics include using a domain generating algorithm to generate random domain names registered to anonymous top level domains such as .xyz. *.top* and *.bid*.

- **Privilege elevation** – Executing malicious system-related activities may require access rights that are beyond those given to the victim's user account. For example, ransomware may want to overwrite the Master Boot Record, which can only be done as an Administrator. Simply asking for administrator access may work or other privilege escalation techniques may be used.

It is important to note that not all ransomware families exhibit behavioural traits from all of the above mentioned categories. Well-equipped ransomware families that achieve wide spread distribution and high success rates such Teslacrypt, generally exhibit most of the mentioned setup traits. However, there is also an abundance of simpler, less equipped ransomware families that exhibit only some of the setup behavioural traits.

To demonstrate this point, refer to Table 1 which shows the composition of setup traits for various ransomware families. Notice variability in how different ransomware families set up. Some families exhibit all traits of a malicious setup (Teslacrypt, CryptoWall) while others only a few (Alma, Crypt2). The variability in setup behavioural traits among ransomware families may prove challenging for behavioural-based detection, especially for detecting simpler less equipped ransomware that exhibits low-profile behavioural setup.

Analysing the setup behaviour of unknown executables is a powerful tool in ransomware detection that can aid in early detection to ensure minimal file loss. Behavioural-based ransomware detectors that negate setup behaviour and only look at data transformation behaviour, may require substantial file encryption to occur before it can confidently flag an executable as ransomware.

Table 1. Behavioural traits of various ransomware families

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

4.2.2 Data encryption

At the heart of crypto ransomware behaviour is its ability to transform mass amounts of data from a usable state to an unusable state. Typically, ransomware data transformation behaviour is defined from a file operation perspective, which gives rise to three categories of ransomware [7]:

Class A ransomware opens the original file and directly overwrites its content with its encrypted data. Class B ransomware first moves the file to a discrete location, encrypts the file as in Class A and then moves the file back to its original location. Class C reads the original file, encrypts its content, writes the encrypted content to a new file, and deletes the original file.

According to the above classes of ransomware, identifying ransomware by its data transformation behaviour requires at the very least the ability to track file operations, and/or the ability to identify data encryption. Such approaches to ransomware detection have proven effective such as in [8], which tracks

the file operations using I/O file requests, and in [7] which detects mass file encryption using statistical measures such as entropy change.

Behavioural-based detection methods primarily based on detecting mass file encryption may be effective, however may come at a resource intensive cost. Consider for example, encryption measures such as entropy change which requires the file entropy to be calculated for every single write operation executed by an application. Furthermore, these operations need to track file operations for each file separately over the life-span of an observed process. Such an approach may considerably deteriorate disk read and write performance and result in high system load.

Analysing file operation behaviour can also provide more direct evidence to a ransomware attack in the form of identifying known ransom notes or known ransomware file extensions. To facilitate the victim in paying the ransom amount, the ransomware needs to make known a ransomware attack has occurred and provide instructions on how the victim's data can restored. This is usually done by writing out ransom notes, often at multiple locations in the user's directory and in multiple formats.

Thus, one way to detect ransomware would be to detect dropped ransom notes. Such an approach requires static analysis on suspicious files to detect ransom related content. For example, one could use a-bag-of-words approach (such as those used in spam detection) to find correlations between phrases found in ransom notes (such as "pay ransom", "encrypted", "Bitcoin") to those in an unknown dropped file.

Identifying dropped ransom notes may be a successful technique for a large group of ransomware families, however it cannot solely be relied upon. Ransomware developers can easily bypass such a detection system by using alternatives to text-based ransom notes. Examples of this include using an executable that displays a window showing the note, changing the desktop background to a picture of the ransom note or inserting one line ransom demand in the file name of the encrypted files.

Ransomware developers that are aware of data transformation analysis techniques have been known to use simple tricks to mask the presence of mass file encryption. These tricks include: only partially encrypting files (such as header sections), or saturating the system with low entropy file write operations, to produce a lower encryption footprint.

In addition to performing write, move and delete operations, ransomware also tends to rename the encrypted file. This renaming is often done to alert the user that a ransomware attack has occurred. An example of file renaming is to appending an additional extension to the original extension such as *example_file.docx* to *example_file.docx.encrypted.* A cost effective solution then to detect ransomware would be to analyse file rename operations to identify ransom-like file names or extensions. Such a technique may work on simple ransomware, but will not work on more intelligently written ransomware such as CryptXXX which randomizes the file name or Spore which retains the original file name. If such approaches are used in isolation, the false-positive rate is usually high.

To summarise, extracting behavioural features associated with data transformation and file operation can provide valuable information necessary to detecting ransomware. However, it cannot be solely relied upon.

4.3 Machine-learned behavioural-based detection

A behavioural approach to ransomware detection requires a decision making algorithm that accepts a quantitative behavioural trace of a running process as an input, to output a simple binary decision – yes it is safe/benign or no it is ransomware.

The field of algorithmic decision making is vast, giving us numerous approaches to solving complex problems. For ransomware detection, we have determined that a supervised machine learned approach is best suited to such a problem.

Supervised machine learning is the optimising of a prediction model to best describe the mapping from the input data to its assigned target label, with the end goal of being able to predict the target label of unknown data. The optimising of the model is alternatively referred to as model fitting or model training.

Explained in the context of ransomware detection, supervised-machine learning is the training of a decision algorithm to recognise certain behavioural traits (the input data) of running processes that optimally discriminates between ransomware and non-malicious benign applications (the target label).

The training of the decision algorithm requires three crucial ingredients: the training dataset containing examples of known (i.e. labelled) ransomware and known benign applications, the capturing of behavioural traits in a quantifiable manner, and the classification scheme which defines the training and prediction algorithm.

4.3.1 Dataset construction

The construction of the training dataset is arguably the most important, yet often times an overlooked task when designing a prediction model. The dataset, if constructed correctly, should be fully representative of the target population to ensure the model is trained on examples that are expected in real-world applications. This is easier said than done for classification tasks such as ransomware detection, where the target population comprises an almost limitless and ever growing collection of software – both benign and ransomware.

For a representative collection of ransomware samples, it is not so much quantity that's important but rather variety – for a noise insensitive prediction model, training on 1000 Locky ransomware samples should prove no more useful than training on just one Locky sample. The prediction model used in RansomFlare was trained on approximately 300 unique ransomware families and variants. Each ransomware sample in the training set was painstakingly analysed and manually labelled by family and variant to ensure a balanced within-class representation of ransomware. To further increase variability in the training set, synthesising techniques were employed to synthesise "future" ransomware behaviour.

The benign dataset serves as the reference point to what is considered ransomware, and thus is equally as important as the ransomware dataset. The bulk of benign executables used in RansomFlare's training was collected from a real-life office of computers with data collectors on them. In addition to these real-life examples, a controlled benign dataset was collected that comprised of specific benign examples that are similar to ransomware. Examples of this include bulk file compression using Winzip or bulk image processing using Inkscape. In doing so, the risk of false-positives is kept to a minimum.

4.3.2 Behaviour capturing and feature extraction

The behaviour of a running executable can quite easily be described linguistically: for example, "the unknown process spawned multiple threads to rapidly enumerate directories". A mathematical model however, requires a concise numeric description.

Many approaches can be taken to collect process behavioural information such as Windows audit logs, event tracing, kernel drivers and process hooking to name a few. These methods generate a wealth of data, however not all data is relevant for ransomware detection. Extracting only relevant information is an art and generally constitutes the bulk of work for a machine-learning task.

The team behind RansomFlare was able to design a highly compact feature set by selecting feature extraction techniques that best quantify the behavioural traits of a malicious setup and malicious encryption. By using a compact feature set, the computation resources are kept low, allowing rapid real-time detection without loading the system.

Although the exact features used in RansomFlare cannot be divulged, a reduced dimensionality feature space, using principle component analysis (PCA), can illustrate the highly discriminatory abilities of RansomFlare's features. Figure 1 shows benign samples (blue dot) and ransomware samples (red dot) at various time steps in this reduced dimensionality feature space. The well-defined clustering of benign and ransomware samples, and more importantly the well-defined separation between these clusters, prove that RansomFlare features are able to effectively discriminate between benign and ransomware behaviour.

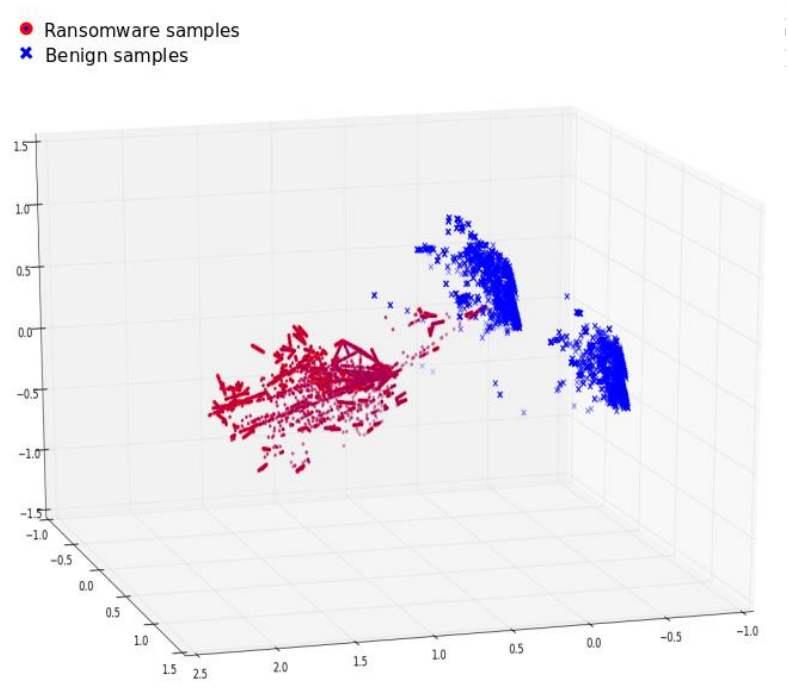

Figure 1. Reduced feature space of ransomware samples and benign samples as seen by the RansomFlare detection model

4.3.3 Classification scheme

Multitudes of well-founded supervised machine learning architectures exist: boosted decision trees, naïve Bayesian classifiers, neural networks and support vector machines (SVM) just to name a few.

Choosing a classification scheme relies on insight into the problem domain to determine the requirements of the classifier. For behavioural-based ransomware detection, three critical requirements have been identified:

- **Real-time operation** – the classification scheme must operate in real-time, with continuous decision making over the life-span of the executable.
- **Resource light** – The prediction algorithm must not load the system resources. Ransomware detection is an important task that would form part of a greater threat detection system and thus should have a minimal resource footprint.
- **Easily updateable** – As new ransomware families appear, the classification scheme must be easily updateable.

Empirical testing revealed that a SVM classification scheme was best suited for ransomware detection as it satisfies the aforementioned requirements. A SVM is a highly effective binary prediction model that uses a sparse training data set (the support vectors), allowing for computationally light, real-time predictions. Its true power lies in its ability to implicitly perform a non-linear feature space transform, making it highly adaptable to a range of various classification problems.

As only the support vectors are required by the prediction algorithm of a SVM - the support vectors being the training samples used to define the decision making plane - newly discovered ransomware samples need not necessarily be added to the training set for model updating. If the new sample is located far from the decision plane, it can be assumed it will not form part of the support vector set and thus the model does not need to be retrained with the new sample. This is hugely advantageous to ransomware detection where new ransomware is discovered daily because it is likely that the model will not require updating.

4.4 Behavioural-based detection limitations

Although far more difficult to implement and less common, behavioural obfuscation techniques have been used by malware developers to attempt to mask the malware's underlying malicious behaviour. These techniques introduce behavioural noise. Such noise is typically introduced on a system call level by using techniques such as randomly ordered system call insertion (i.e. inserting redundant system calls), system call substitution (i.e. using alternative but comparable system calls) or system call reordering (i.e. randomising the order of system calls where ordering is irrelevant).

A robust behavioural-based malware detection solution should be one whose performance is invariant to system noise. This system noise should include general noise introduced by variability in development techniques as well as purposeful noise introduced by obfuscation techniques.

RansomFlare uses high level behavioural features that have shown to generalise behaviour well even under the presence of obfuscation noise. This is evident when tested against the polymorphic ransomware Virlock, where RansomFlare generates the same behavioural feature set over multiple execution runs of Virlock samples.

5. Case study: CryptoLuck ransomware

To highlight the importance of behavioural-based ransomware detection, consider the case study of the CryptoLuck ransomware.

First seen near the end of 2016, the CryptoLuck ransomware (a unique variant of the CryptoLocker family) was distributed using malvertising and the RIG-E exploit kit. Victims got infected by visiting a site that used a RIG-E exploit kit to deliver a ransomware payload. The CryptoLuck payload was discretely downloaded and executed on the computers of unlucky internet browsing users.

CryptoLuck is an interesting case study as the payload comes in three parts, which are contained in a self-extracting archive: a legitimate Google update executable *GoogleUpdater.exe*, a legitimate accompanying configuration file *crp.cfg*, and a malicious DLL file *goodate.dll*.

During the execution of the legitimate Google updater, the executable loads the DLL file *goodate.dll* located in its current directory. CryptoLuck exploited DLL hijacking techniques and replaced the legitimate Google DLL with its own malicious DLL. This malicious DLL once loaded, will unpack and execute the ransomware payload.

By using a legitimate signed executable to unpack the ransomware, CryptoLuck was able to execute undetected from major commercial virus scanners. Possible reasons for this being that virus scanners have been known to automatically whitelist signed executables and their child processes.

Using DLL hijacking as a technique for anti-detection is not effective against RansomFlare. This is because it is designed to treat all user space executables (signed and not signed) as potential ransomware threats. RansomFlare was able to flag the ransomware related child executables of *GoogleUpdater.exe*, even though it did not form part of the ransomware training set (previously unseen). This is because the application exhibited classic behavioural traits that could only be associated with ransomware. Some of these traits include:

+ **Payload persistence** – The CryptoLuck inserts a registry run key entry to ensure *the GoogleUpdate.exe* executable resumes on system reboot.

+ **Stealth techniques** – The GoogleUpdate.exe and accompanying DLLs are stored in the %AppData% directory where it is executed in stealth. The ransomware uses DLL hijacking of a signed executable to masquerade as a legitimate executable.

+ **Environment mapping** – The ransomware determines if it is running in a virtual environment by searching for VM related keywords such as *vmbox*, *vmware* and *wine* in the registry keys.

  The ransomware maps out mounted drives and unmapped network shared drives and cloud-based drives. Only specific files relating to user data such as docx, xlsx, pdf, pptx, jpeg, jpg undergo data transformation.

+ **Data transformation** – Targeted files are encrypted with a solid encryption routine: for each targeted file, an AES-256 key is generated and used to encrypt the file. An embedded RSA public

key is used to encrypt the newly generated AES key and then stored in the encrypted file. All encrypted files undergo a name change as an <victim id>_luck extension is appended to the name.

During encryption, a ransom note in the form of a text file is dropped in every targeted folder location. This ransom note is named @WARNING_FILES_ARE_ENCRYPTED. <victim id>.txt. When all files are encrypted, the desktop background is changed to a ransom note and a window is launched with a ransom note. See Figure 2 for a screenshot of a CryptoLuck infected computer which shows the three forms of ransom notes.

The encryption mechanism of CryptoLuck is solid, and as such, no decryption tool has yet to be made available by cybersecurity researchers. An infected user would need to pay the required 2.1 Bitcoins, or recover from backups to restore his or her data.

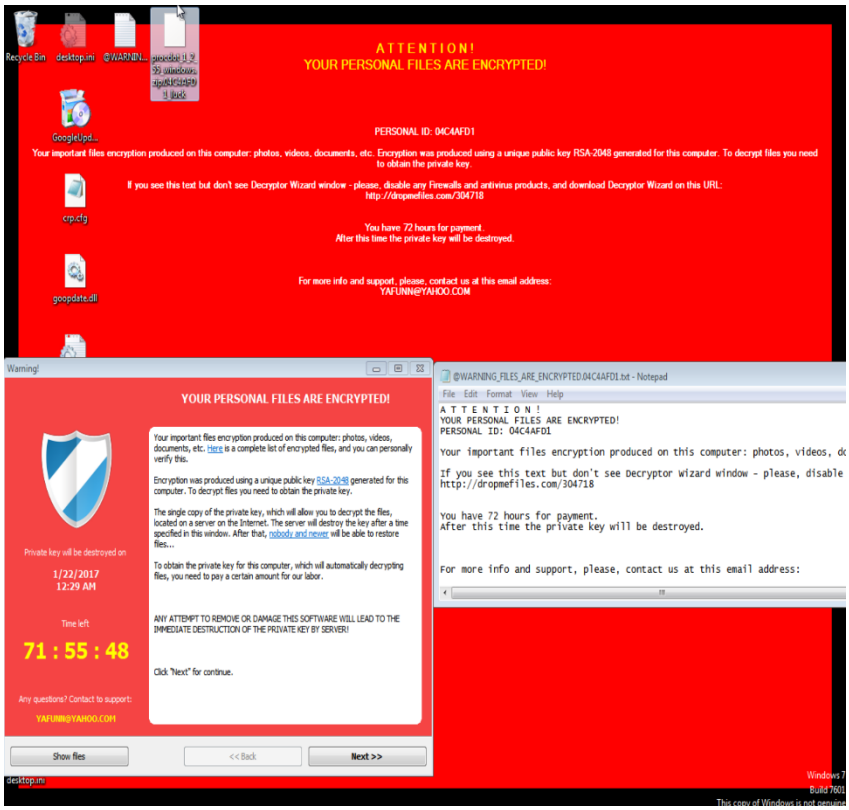

Figure 2 Screenshot of a computer that has been infected by a CryptoLuck ransomware attack.

6. Looking to the future

As long as infected users are willing to pay for their ransomed data, the underground ransomware industry will continue to strengthen, finding new and creative ways to outsmart common prevention mechanisms. To highlight the continued innovations of the ransomware industry, consider the following predicted trends:

+ **Worming capabilities** – ransomware will look to rapidly compromise entire network systems, propagating in a computer-to-computer fashion.

+ **Targeted attacks** – it has been proven that businesses are willing to pay ransom demands to quickly restore business operations. It is predicted that ransomware will be used in a more targeted fashion by compromising single endpoints, moving laterally through networks and manually ensuring that ransomware is executed on critical assets. This will ensure that the targeted organisation pays the ransom demand and it could be much larger than typical "spray-and-pray" approaches.

+ **Secondary payloads** – Ransomware bundling secondary malware for increased firepower. Expect other monetisable attacks to be bundled with ransomware as the creativity of syndicates continues.

+ **Attacks against non-traditional systems** – Extending the concept of ransomware to non-conventional computing devices. More and more technology is connecting to the internet and is starting to attract the attention of the malware community. Ransomware targeting mobile devices and IoT systems already exist and this trend can be expected to increase.

The best that the cyber security community can do going forward is to provide smarter detection techniques capable of predicting new ransomware. This should be in conjunction with a layered security model. This layering could consist of network filters (e.g. spam filters), static-based detection and behavioural-based detection. RansomFlare represents the last line of defence and provides behavioural-based detection, which is effective against ransomware.

7. Conclusion

As evident by the uncontained growth of ransomware over the last few years, signature based detection techniques have proven an ineffective defence.

Static-based detection is effective against known ransomware, however the continuous influx of new ransomware proves difficult to detect on an acceptable time scale. Furthermore, static obfuscation - particularly in the form of malware factories - are being used to avoid detection of known ransomware.

A more effective ransomware detection scheme is one that has predictive capabilities to make intelligent threat inferences of unknown processes. This can be achieved by treating all running executables as unknowns, where the threat level is continuously updated based on how the executable is behaving. RansomFlare uses such an approach by using dynamic (behaviour) analysis in conjunction with machine learning to provide predictive abilities capable of zero-day ransomware detection.

8. References

[1] “Trend MIcro,” 6 Nov 2016. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/research-and-analysis/predictions/2017. [Accessed 2 2 2017].

[2] C. Online, “Report: Only 3 percent of U.S. companies pay attackers after ransomware infections,” 16 Feb 2017. [Online]. Available: http://www.csoonline.com/article/3101863/security/report-only-3-percent-of-u-s-companies-pay-attackers-after-ransomware-infections.html.

[3] “Trend Micro,” 8 Jun 2016. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-attack-on-university-of-calgary-forces-20000-payment. [Accessed 2 2 2016].

[4] P. Belcher, “Sofos - Invincea,” 2016 Jun 2016. [Online]. Available: https://www.invincea.com/2016/06/hash-factory-new-cerber-ransomware-morphs-every-15-seconds/.

[5] K. You and I. Yim, “Malware Obfuscation Techniques: A brief survey,” in *Internation conference on Broadband, Wireless Computing Communication and Application.*, 2016.

[6] L. Robert, “We Live Security,” 22 December 2014. [Online].

[7] K. Butler, N. Scaife, H. Carter and P. Traynor, “CryptoLock (and Drop it): Stopping Ransomware Attacks on User Data,” in *Internation Conference on Distributive Computing Systems*, 2016.

[8] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge and E. Kirda, “Cutting the Gordian Knot, a Look under the Hood of Ransomware Attacks,” *Lecture Notes on Computer Science,* vol. 9148, pp. 3-24, 2015.