Detecting Cryptographic Ransomware by Examining File System Activity

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Meltem AKAY

in partial fulfillment for the
degree of Master of Science

in
Cybersecurity Engineering

İSTANBUL
ŞEHİR
UNIVERSITY

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Cybersecurity Engineering.

APPROVED BY:

Prof. Ensar Gül ....
(Thesis Advisor)

Dr. İsa Sertkaya ....
(Thesis Co-advisor)

Asst. Prof. Erdinç Öztürk ....

Asst. Prof. Mehmet Serkan Apaydın ....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 29.07.2019

SEAL/SIGNATURE:

Declaration of Authorship

I, Meltem AKAY, declare that this thesis titled, 'Detecting Cryptographic Ransomware by Examining File System Activity' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 29. 07. 2019

Detecting Cryptographic Ransomware by Examining File System Activity

Meltem AKAY

Abstract

Cryptographic ransomware, which locks a victim's files and demands payment to re-establish access, is one of the most dangerous and popular cyber crimes of today as it gives attackers a golden opportunity to extort money. Although many different approaches are presented to detect and prevent this troublesome malware, recent research suggests that none of these approaches are flawless and they can be bypassed.

In this thesis, we propose CRYPTOCOP, a protection system that stops a ransomware attack in the early stages. The defense mechanism limits the applications' capability of executing file write functions, which is excessively performed by a typical ransomware. We define an adaptive threshold mechanism for file write requests of each running process, which facilitates benign file system operations while terminating the malicious activity. The results of experiments show that CRYPTOCOP is able to stop 706 out of 736 (~96%) ransomware samples with minimal loss of files – less than 5 – and a negligible performance overhead.

Keywords: Ransomware, Malware, Engineering, File System Examination, User Data Protection

Dosya Sistemi Aktivitelerini İzleyerek Fidye Yazılımların Tespiti

Meltem Akay

Öz

Kriptografik Fidye Yazılımları, üzerinde çalıştığı sistemdeki dosyaları gelişmiş kriptografik yöntemlerle şifreleyen ve kullanıcının tekrar erişim sağlayabilmesi için fidye talep eden yazılımlardır. Saldırganlara, kullanıcılardan kolayca para koparmayı sağlaması sebebiyle günümüzün en popüler ve tehlikeli siber tehditlerinden biridir. Kriptografik fidye yazılımlarını yakalamak ve durdurmak için çeşitli yöntemler sunulmuş olmasına rağmen son yapılan araştırmalarla beraber, bu yaklaşımların atlatılmak için kullanılabilecek hataları bulunmuştur.

Tez kapsamında, kriptografik fidye saldırılarını başladıktan kısa bir süre sonra durduracak bir yaklaşım, CryptoCop, sunmaktayız. Kriptografik fidye yazılımlarının ayırt edici özelliği aşırı dosya yazma işlemi olması sebebiyle, tanımladığımız korunma mekanizması çalışan uygulamaların dosya yazma isteklerini sınırlamaktadır. Belirlediğimiz eşik değeri iyi huylu uygulamaların çalışmaya devam etmesine olanak verecek şekilde uyarlanmaktadır. Elde ettiğimiz verilere göre, CryptoCop ihmal edilebilir performans kaybı ve az sayıda dosya kaybı ile kötümcül yazılımların bu türünü durdurmayı %96 lık bir oranla başarmaktadır.

Anahtar Sözcükler: Fidye Yazılımları, Zararlı Yazılım, Dosya Sistemi Gözetlenmesi

Acknowledgments

I would like to thank my thesis supervisor Prof.Dr. Ensar Gül and my co-advisor Dr. İsa Sertkaya for their guidance. I would also like to thank my committee members for taking out time from their schedule and attending my dissertation.

I owe my gratitude to Ziya Alper Genç. I would like to thank him for his helpful advice on various technical issues examined in this study.

Thank you, Anil, for being in my life. And this work would not have been possible without your help and support.

I wish to give special thanks to my sister Özlem, my brother Erdem and my cousin Serpil for enjoying the life together with me.

The greatest thanks are to my parents, for their support throughout my whole life.

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

API	**A**pplication **P**rogramming **I**nterface
ASLR	**A**ddress **S**pace **L**ayout **R**andomization
AV	**A**nti **V**irus
C&C	**C**ommand-and-**C**onquer
CPU	**C**entral **P**rocessing **U**nit
CSPRNG	**C**ryptographically **S**ecure **P**seudo **R**andom **N**umber **G**enerator
DEP	**D**ata **E**xecution **P**revention
DLL	**D**ynamic **L**ink **L**ibrary
DNS	**D**omain **N**ame **S**ystem
IAT	**I**mport **A**ddress **T**able
I/O	**I**nput/**O**utput
IP	**I**nternet **P**rotocol
KVM	**K**ernel-based **V**irtual **M**achine
LCM	**L**east **C**ommon **M**ultiple
NAS	**N**etwork **A**ttached **S**torage
OS	**O**perating **S**ystem
RaaS	**R**ansomware-**as**-**a**-**S**ervice
SDN	**S**oftware **D**efined **N**etworking
SMS	**S**hort **M**essage **S**ervice
SSD	**S**olid **S**tate **D**isk
SSDT	**S**ystem **S**ervice **D**ispatch **T**able
VGA	**V**ideo **G**raphics **A**rray
VM	**V**irtual **M**achine

Chapter 1

Introduction

Ransomware is a type of malware that locks a victim's data and demands money to re-establish access. Based on the locking method used, ransomware can be categorized into two groups: *computer locker* and *cryptovirus* [35]. The former locks the functions on the target computer system while the data remains intact. The latter, however, encrypts the victim's data using modern cryptographic techniques, e.g., the infamous CryptoWall [7] family. The scope of this thesis is the second category which is also known as *cryptographic ransomware*, or shortly *ransomware*.

Once infected, ransomware uses strong cryptography for encrypting user's files on both local and network attached storage devices and demands a reasonable ransom (or higher amounts in some cases) in exchange for the decryption keys [23]. The payment is asked using cryptocurrencies which makes it harder to be traced by the authorities. In order to increase their revenue, cyber-criminals try to propagate the ransomware as much as possible. To this end, ransomware primarily targets Windows operating system (OS) [2, 43], as it has the largest market share on desktop computers [38]. For instance, WannaCry authors exploited a zero-day vulnerability in Windows OS and performed a global attack [48] (more details about numerous incidents will be discussed in Section 2). Security intelligence companies warn that the dimensions of the ransomware threat increase every day. The potential targets include individuals, public institutions and enterprises. It is expected that a business will be attacked by ransomware every 14 seconds by the end of 2019 [29]. Apart from its economical damage, affected critical infrastructures or public services may stop functioning and lose their reputation [30].

1.1 Problem Definition

Ransomware detection and termination is a challenging task for traditional anti-virus (AV) solutions because of the following two reasons. Firstly, ransomware with metamorphic capability [49] can alter itself, resulting an executable with a different signature. Consequently, it is unfeasible to keep the signature databases of AV vendors up-to-date. Secondly, ransomware's basic cryptographic operations such as key generation and encryption are very common methods, which are also performed by benign programs such as web browsers, password managers, office applications and zip utilities in addition to legitimate cryptographic applications. As a result, limited heuristics capability dramatically increases the chance of false positives and damages the user experience.

Anti-ransomware solutions are developed with the purpose of dealing specifically with the ransomware threat. That said, these approaches focus on today's variants, however, ransomware is getting more sophisticated over time [44], as it happens with the other malware types. Furthermore, existing techniques for ransomware detection are insufficient to tackle the problem due to various limitations, such as, key escrow being sensitive to obfuscation and behavioral analysis being evaded by adaptive attack techniques [12]. Therefore, we propose an approach that tackles the ransomware problem in a more fundamental way.

1.2 Methodology

The thesis starts with literature survey to learn the details of Ransomware and examines previous work that detect various families of malware in infected computers. After examining general behaviour of ransomware, the distinguishing characteristic which is *excessive file write requests* was considered as worth to experiment as an indicator. Our initial expectation was that it would give us the advantage of combating ransomware without affecting system response time.

After deciding the approach, following steps are taken in order to evaluate the approach:

- A test environment, a hardened 32-bit version of Windows 7 OS virtual machine, is created. The OS selection has been based on the fact that Windows platform being the main target of ransomware [2], and version 7 being one of the most widely used edition of the OS [39]. That said, the proposed technique in this thesis can be easily adopted to the other operating systems such as Linux or macOS.
- Ransomware samples are collected and filtered active samples in order to test the functionality and effectiveness.

- A prototype, called CryptoCop, which is a dynamic linked library intercepting Windows File Application Programming Interface (API) functions calls, evaluating the activities and terminating suspicious processes is implemented.
- The effectiveness of the prototype is evaluated over 736 active ransomware samples.
- The usability of the prototype is evaluated over common applications.
- Performance experiments are run to measure the overhead added by our prototype.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter 2 contains the characterization of ransomware and its history. Chapter 3 includes the previous work related to detection methods. In Chapter 4, we discuss our approach design principle and CryptoCop architecture in detail. Chapter 5 gives the details of evaluation process of the approach. The prototype implementation details, test environment setting up, experiments, tools used in the process are described. Chapter 6 details our finding about detection capability, usability and performance. Chapter 7 discusses the limitations of the approach and the prototype, and compares to the previous works. Finally, Chapter 8 contains our conclusion based on our findings.

Chapter 2

Background

This chapter provides background information to the readers that are not familiar enough with Ransomware about classification and evolution process of ransomware.

2.1 Classification of Ransomware

Ransomware is a type of malware that locks the access and demands money to re-establish access. Based on the locking method used, ransomware can be categorized into two groups:

1. *Computer Locker*: locks the functions on the target computer system via heavily consuming system resources while the data remains intact.
2. *Cryptovirus*: encrypts the victim's data using military grade encryption.

Even though both types have the same objectives, their approaches are quite different [35]. The scope of this thesis is the second variant which is also known as *cryptographic ransomware.*

A typical execution of cryptographic ransomware can be exemplified as follows. After infected, cryptographic ransomware immediately gets encryption keys. Then, it scans directories to find files with predefined extensions and tries to encrypt them silently. It aims to catch any valuable data which the victim would willingly pay the ransom to retrieve. When the encryption phrase is completed, an information text file is placed in every affected directory. It also displays a warning banner with a countdown timer. The banner informs the victim that her/his files has been encrypted and all data will be destroyed if they do not pay the ransom in the given time. However, generally the infected computer remains functional.

Ransomware is generally developed with hybrid cryptosystem to be more secure, which is the system providing data protection with the symmetric algorithms and key protection with asymmetric algorithms. Thus, ransomware generally creates symmetric keys on a victim's computer and encrypts files, then encrypts these symmetric keys with the public key.

Ransomware generally follows a sequential approach for file processing. In other words, it encrypts plain-text data, stores the encrypted data, removes the original file and then moves on the next file. The main motivation points for processing this way are as follows: (i) Lower code complexity (ii) Minimal disk usage (iii) Increasing the chances of having some valuable files encrypted in case getting detected before encrypting all data

Ransomware can be classified into three groups regarding their way of processing files [36]:

1. This group of ransomware overwrites the file with the encrypted form of the original data. Some of them also change the name of the file.
2. This type first moves the file to the another directory, reads the data and writes having encrypted data to the same file. Lastly, it moves the file to the previous folder. They might also change the original file name.
3. This group creates a new file and writes encrypted data after reading and performing encryption and then deletes the original file. Different than the rest, this type uses two different stream to write and read operations.

Ransomware uses various methods for propagating including social engineering, spam email, short message service (SMS) message, downloader, data breach, exploit kits, bot infections. Even some cyber criminals specialized in this area are paid in return for defeating the strong defenses and distributing the malware.

2.2 The Evolution Process

The history of ransomware starts with the `AIDS Trojan` created by Joseph L. Popp in 1989. The virus was distributed via floppy disk and encrypted file names by using a weak symmetric encryption algorithm [34].

In 2005, `GPCode` or `PGPCoder` was the first modern example of ransomware that encrypts user data on infected computers and demands a ransom for data recovery [35]. Its first variants had some flaws such as weak custom-encryption algorithms, using only symmetric encryption which made the attack revertible easily [15–17]. The enhanced variant called `Gpcode.ax` appeared in late November 2010. The new one used stronger cryptographic algorithms, RSA-1024 and AES-256. Unlike the previous variants, it worked by overwriting the data in files instead of creating new file and deleting the one, which prevented recovery via data-recovery tools [18].

In 2006, `Trojan.Cryzip` attacked victims' data by creating a password-protected ZIP file for each user file with the target extensions [40]. However, it was quickly overcome since the password was embedded inside the source code [35].

`Trojan.Archiveus` also came forward in 2006. Differently, the victims were requested to buy some medications from certain online pharmacies and share the order ID [35] to obtain the password to recover their data.

In 2008, the first locker ransomware called `Trojan.Randsom.C.` emerged. In 2011, it was the large-scale locker ransomware outbreak, 60,000 new ransomware samples were identified. The demanded ransom was around US$150 to US$200 [35] per victim.

Between 2012 and 2014, locker ransomware disguised as law enforcement was used as an effective way to force victims to pay. For example, the version of the `lyposit` toolkit called `Reveton`, notified victims with a pop-up message that their computers have been locked by the FBI or U.S. Justice Department downloading copyrighted material, or due to some other criminal activity [34]. However, these locker ransomware variants could be removed using security softwares [35]. With increasing awareness of the threat and easy recovery methods, cybercriminals' income started to decrease.

In time, cybercriminals focused on cryptographic ransomware again, learned the critical component of encryption and achieved to create successful ransomware.

In November 2013, one of the famous attacks, `CyptoLocker`, was made. It spread via email attachments or an existing infrastructure of `Gameover ZeuS` botnet, encrypted 67 different file types, utilized RSA encryption algorithm and demanded a payment around two Bitcoins equivalent of US$100 at that time within 72 or 100 hours. In May 2014, the variants of CyptoLocker were reported to have infected 500 000 machines and have received ransom from 1.3 percent of victims [34]. In June 2014, to thwart this threat, Cryptolocker distribution servers were taken down by a consortium constituting a group of law enforcement agencies, security software vendors, and several universities [34].

After the success of Cryptolocker, the attackers focused on developing new techniques. The number of ransomware families increased by 250% between 2013 and 2014 [35]. In February 2014, `CryptoDefense` appeared. Despite the poor implementation of the cryptographic functionality, the ransomware earned $34,000 in its first month [41].

`CryptoWall`, the more sophisticated form of CryptoDefense, appeared in April 2014. The family propagates via malicious attachment and links in spam emails and unfortunately its variants are still a threat today [26]. Its third variant had earned more than US$325 million from a mix of end users and businesses alone [7]. `Koler.a` is also launched in April 2014 and infected around 200 000 Android users. Another ransomware detected in 2014 is `SynoLocker` targeting Synology network storage.

In May 2015, Ransomware as a Service (RaaS) appeared, which offers a service for attackers to create, launch and maintain their ransomware without having any technical skills [35]. 342 000 new variants were seen in that year [42].

While new enhanced versions of existing ransomware appeared such as the fourth iteration of CryptoWall [8], new ransomware targeting different OS appeared. In September, `LockerPin` was released to infect Android systems. In November, `Linus.Encoder.1` targeting Linux systems was discovered by Dr. Web, a Russian computer security firm [34].

In 2016, although the number of new variants decreased to 241 000 when compared to 2015, the infection rate increased to 361 000 from 470 000 [42]. Attackers started to use Javascript which allows to perform multi-platform attacks, including Linux and macOS, and infected thousands of WordPress web sites [34].

In this year, many other ransomware families were seen, including `Petya`, `Locky`, `Cerber`, `Sage`, `Mamba`. Additionally, in 2015-2016, the demanded ransom increased by a factor of three from US $294 to US $1077 [42].

In May 2017, the trend changed and the number of variants started to increase. Two important outbreaks were seen in this year: `WannaCry` and `Petya`. They propagated by exploiting critical vulnerabilities (later named as EternalBlue) in Windows OS and infected thousands of computers in more than 150 countries within a matter of hours. Other major ransomware families were `Jaff`, `Globalmposter`, Cerber, Locky, Sage, Mamba. In that year, the average amount of demanded ransom was stabilized at approximately US $544 [42].

The ransomware kept hitting companies and causing financial and reputational damage in 2017. For instance, South Korean web hosting firm Nanaya hit by `Erebus`. The company ended up paying approximately US $1 million but also its stock price fell by over 3% due to the attack. Another incident was Danish shipping giant AP Moller-Maersk being hit by a variant of Petya and declared to lose up to US $300 million [42].

Chapter 3

Related Work

This chapter gives information about previous work related to this thesis work that tries to detect ransomware to gain better understanding.

3.1 Practical Limitations of Backup Strategies

At first glance, one can consider that ransomware can be mitigated by backing-up all critical data regularly and restoring the files in case of an attack. However, this strategy is not an effective and efficient way because of common bad habits such as insufficient back-up frequency, non-comprehensive file coverage, or insecure configuration of back-up locations that can be accessed by ransomware. This has been confirmed by a survey on backup practices [9] which reports that only 42% of users were able to fully restore their data after a ransomware attack.

3.2 Literature Review

Naturally, the growing threat of ransomware gained the attention of security professionals. In the cryptographic literature, there have been several proposals to combat ransomware. These can be classified into six groups regarding their defense strategy: (i) access control; (ii) behaviour analysis; (iii) key escrow; (iv) decoy files; (v) hardware-assisted protection; and (vi) network-level defense.

- **Access Control**: In the context of ransomware defense, this approach protects the system *before* file modification by controlling accesses to the cryptographic functions. For example, UShallNotPass [11] proposes a mechanism that all processes, except white-listed ones, need to get permission from the user to call the critical cryptographically secure pseudo-random number generator (CSPRNG) function. UShallNotPass leverages the necessity of generating secure random numbers to perform strong encryption. Differently, Palisse et al. [32], an earlier proposal, replaces the cryptographic service provider in Microsoft Cryptographic API (MS CAPI) to control built-in encryption functions.

- **Behaviour Analysis**: This approach attempts to detect ransomware *during* the attack by monitoring process behaviors and looks for anomalies, such as aggressive modification of different files types. For example, UNVEIL [20] considers repetitive I/O requests for multiple files as an anomaly. All I/O requests are fingerprinted (process, file, I/O type, write buffer entropy) by and recorded in a sequence. Repeated entries in this list are interpreted as ransomware activity. CryptoDrop [36] monitors modification of file header, change of file's entropy and dissimilarity between file contents before and after a write operation. A process which triggers all these indicators or reaches the threshold is considered as ransomware. ShieldFS [4] looks at low-level I/O activity and searches for excessive file read/rename/write, directory traverse, high entropy write operations and file type coverage. Different from the two previous works, ShieldFS creates shadow copies of all modified files to prevent data loss. Similarly, Redemption [19] uses the same indicators, but applies the write requests to sparse files and commits the changes later. Lastly, DaD [33] calculates the chi-square goodness-of-fit test on the write buffer of processes to detect random looking content which may be a encryption method outcome.

- **Key-escrow**: This approach leverages the fact that the symmetric algorithms must be employed for feasible encryption of files. Obtaining the secret keys used in the victim's computer would therefore enable to recover the encrypted files. In this regard, [24] and PayBreak [22] intercepts crypto API calls and logs the parameters in a secret vault. After the attack, the correct keys are searched by the brute-force method within this vault. In a slightly different method, [21] proposes to put a backdoor in CSPRNG of the host to reproduce the keys generated by ransomware and recover the files.

- **Decoy Files:** Another strategy to detect cryptographic ransomware is to employ decoy files to detect malicious file system activity. In this strategy, carefully-crafted fictitious files are placed in the file system among real files. The user is informed about the decoys and is not supposed to write on them, so any write request to the decoy files is treated as an indicator of ransomware activity. RWGuard [27] uses this technique –in addition to behavioral analysis– to mitigate ransomware threat in real time. Moreover, there are commercial applications, e.g., CryptoStopper [47] which use decoy files to detect cryptographic ransomware.

- **Hardware-assisted Protection:** Different from software-based solutions, FlashGuard [13] is a hardware-level anti-ransomware that leverages the capabilities of Solid State Disks (SSDs). To protect the data, FlashGuard patches the device's firmware and modifies the garbage collection mechanism of the SSD to keep the copies of the data encrypted by ransomware. When the victim is aware of the ransomware attack, the files can be recovered by using the copies available on the disk.

- **Network-level Defense:** An alternative protection strategy is to mitigate ransomware at the network level. Certain ransomware families download encryption keys from a remote location, i.e., Command-and-Conquer (C&C) servers. If this communication is disrupted, i.e., the malicious server's IP is blacklisted and the network firewall blocks the connection, the keys cannot be delivered. Towards this goal, [3] proposed a software-defined networking (SDN) based approach to block C&C servers after a threat is detected. Furthermore, [6] employed machine learning techniques and developed an SDN-based system to prevent delivery of keys on HTTPS traffic which malware is shifting towards to utilize.

Chapter 4

CryptoCop Design

We proposed CryptoCop as an early warning system to detect and stop the malicious activity of cryptographic ransomware. In order to present our approach in detail, this chapter begins with design principles that are considered while mitigating ransomware. Next, the prototype, called CryptoCop, security assumption and architecture are described.

4.1 Design Principles

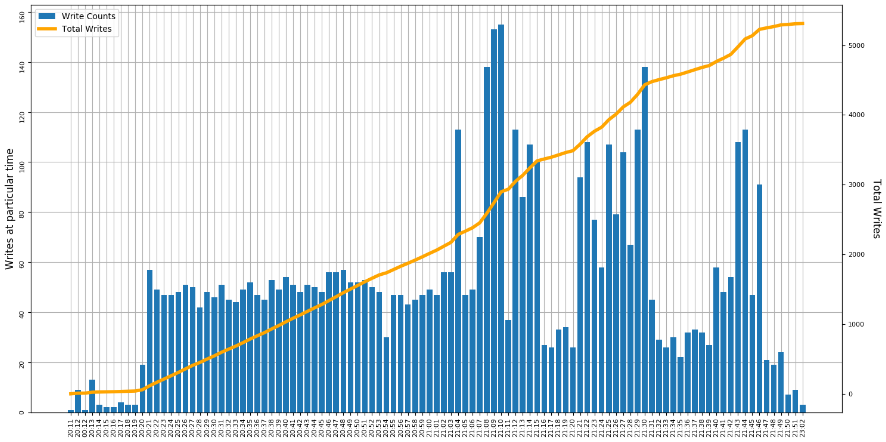
We developed our defense method by not only using the observations on the behaviour of a wide set of cryptographic ransomware families but also considering the attack surface extension possibilities.

To the best of our knowledge, the requirements of a successful ransomware campaign with a mass impact is first analyzed in detail by [10].

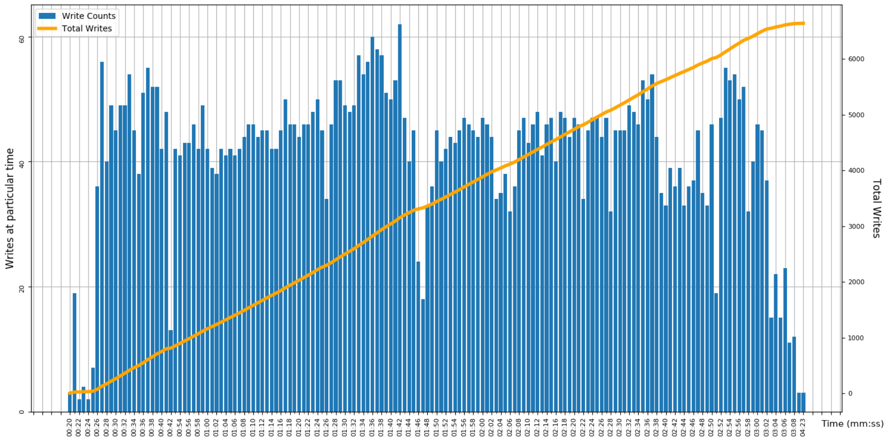
Among others, it is crucial to note that cryptographic ransomware needs to perform two fundamental tasks to achieve its nefarious aims:

- **Encrypt plaintext data.** Access to original data should be locked using a reversible mechanism which can be performed by only the attacker. Cryptographic ransomware employs encryption algorithms to lock access to the victim's data.
- **Destroy plaintext data.** The original data must be inaccessible, i.e., deleted or overwritten, in order to force the victim to pay the ransom.

As we reviewed the previous work in Chapter 3, cybersecurity community proposed efficient defense systems to address the ransomware threat. However, the combat against



(a) Write activity of a TeslaCrypt sample



(b) Write activity of a Bitman sample

FIGURE 4.1: The bars represent the number of unique files modified by ransomware at given time point. The yellow line shows the number of written files at a specific time.

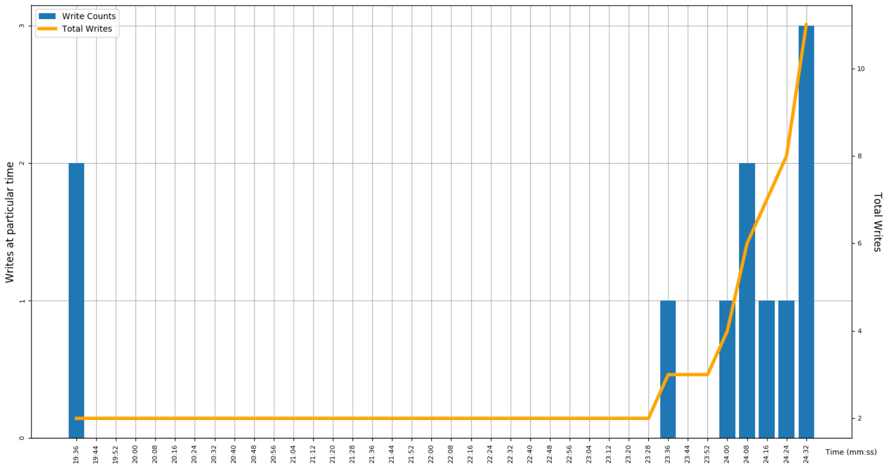
malware is a never-ending war, and the history suggests us that the ransomware, although today mostly fairly contained, will evolve to bypass current defenses. For example, [12] showed that next generation ransomware may use alternative techniques to evade detection. It should be noted that the damage of ransomware might be irreversible, especially when the encryption methods are used properly. We designed CRYPTOCOP to be ready for this potential threat.

The following principles are considered when designing CRYPTOCOP.

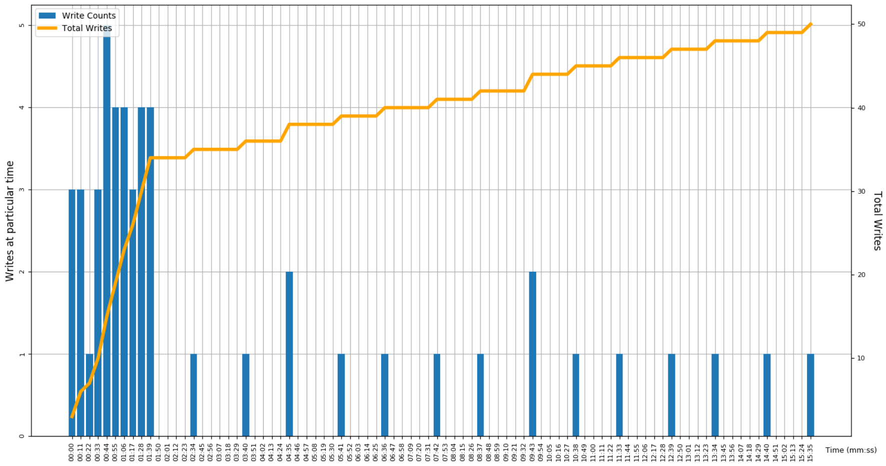
- **Statistical anomaly indicators can be evaded.** Behavioral analysis-based defenses try to detect ransomware activities by looking for the statistical anomalies by utilizing chi square goodness-of-fit test, entropy, etc. If a ransomware finds a way that does not trigger such indicators, it would not be detected, see [12] for instance. Therefore we do not adopt statistical measures in our defense strategy.

- **Controlling crypto APIs can be nullified.** There are long-tested and publicly accessible cryptographic libraries that can be utilized by ransomware. In addition, ransomware can leverage obfuscation which makes intercepting/controlling these operations unfeasible. Consequently, we abandon this strategy and do not use in the defense system.

In the light of these tenets, we recognize only one indicator of ransomware that cannot be hidden or substituted: *aggressive increase in the calls to file write API* of the host OS.Figure 4.1 distinguishably displays the dense write activity of two ransomwares, namely TeslaCrypt and Bitman, as compared to activity of benign application in Figure 4.2.



(a) Write activity of compression and decompression of 5 files (1GB) with WinZip



(b) Write activity of downloading a torrent of 22 files(3GB) with µtorrent

FIGURE 4.2: The bars represent the number of unique files modified by ransomware at given time point. The yellow line shows the number of written files at a specific time.

4.2 Security Model and Assumptions

As a defense system, CryptoCop aims to protect the host from cryptographic ransomware. Non-cryptographic ransomware variants are beyond the scope of this paper. We also consider that the ransomware is able to encrypt victim's data using any algorithm. That is, we do not make any assumption on the encryption technique.

We designed CryptoCop as a software module running on the environment provided by the host OS. As a natural consequence, we assume that the host OS prevents vertical privilege escalation, otherwise the attacker may obtain administrator rights and disarm CryptoCop. It should be noted that, even if the host OS is up-to-date, an attacker with the knowledge of a zero-day vulnerability may disable the protection.

Furthermore, no modern OS allows direct disc access, i.e., processes need to call the specific APIs provided by the host OS in order to write data to disk. For example, on Windows platform, user mode processes invoke `WriteFile` API for disk I/O. Therefore, we assume that ransomware must use system APIs to write encrypted data back to disk.

4.3 Architecture

In essence, CryptoCop is a file system monitor which tracks the most fundamental activity of ransomware: writing (encrypted) data. By controlling the calls to file write API, CryptoCop maintains a malice score for processes and allows only the normal-behaving applications to write to disk. Otherwise, for instance, if an application goes beyond the security threshold, it is terminated immediately. Figure 4.3 depicts the workflow of CryptoCop.

4.3.1 Controlling File Write API

User-mode processes need to use APIs provided by modern OS to write data to disk. Since these file write functions are well-known and their number is limited, it is feasible for CryptoCop to monitor and control them.

4.3.2 Malice Score Computation

CryptoCop maintains a malice score for each process, based on the file write activity. It should be noted that malice score is a value maintained individually inside each process hence no additional synchronization is required among processes.

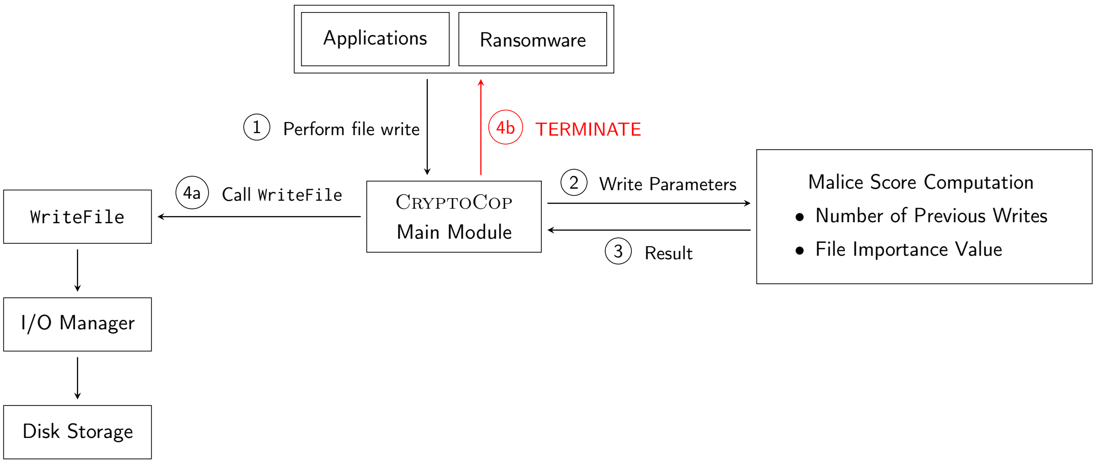


Figure 4.3: Design overview of CryptoCop. The module intercepts file write calls and updates the malice score accordingly. If the malice score of a process exceeds the threshold, the related process tree is terminated. Otherwise, the call is dispatched to the I/O Manager.

Let F denote the set of files and $F_p \subseteq F$ be the files modified by the process p. Malice score of p, denoted M_p, is the sum of importance values, v_f, assigned to the modified file f for all $f \in F_p$. Thus, we have

$$M_p = \sum_{f \in F_p} v_f \tag{4.1}$$

For all processes, M_p is recalculated after each intercepted call to `WriteFile` API if the modified file f is being accessed for the first time by the process p. If M_p exceeds T, where T is the security threshold, CryptoCop blocks the call and terminates p. Otherwise, the call is forwarded to the I/O manager and control is routed to p.

4.3.2.1 Determining Security Threshold

The values for T and v_f are configurable and have to be assigned according to the users' requirements. The main motivation behind this flexibility is to capture various use cases that can arise from the fact that users have different number of files and file types, and different level of importance per each file etc. For instance, while an office computer containing confidential data needs to keep more files safe, a commodity user might tolerate losing a larger number of files. Furthermore, highly critical files such as databases can even be assigned importance values higher than the threshold in order to make sure only database user can modify them. However, it needs to be underlined that setting v_f too strict will cause an increase in the rate of false positives. In other words, the strictness of these parameters is a trade-off that has to be made by the user between usability and accepted risk of file loss.

Chapter 5

Evaluation Process

This chapter discusses development and evaluation process of our solution. In other words, it can be called as our road map. It starts with giving information of our prototype, CryptoCop, implementation details. The important steps of experimental environment set-up are listed. Finally, the critical tools used in the development process are mentioned.

5.1 CryptoCop Implementation

To evaluate the effectiveness of the selected indicator, we developed a prototype[1] of CryptoCop. On Windows platform, applications call `WriteFile` API to write data to the disk. Therefore, our prototype intercepts each call made to `WriteFile` API and computes the malice score of the caller process in real-time. To accomplish this goal, our CryptoCop implementation comprises two modules:

- Main Module, which intercepts the invocations of `WriteFile` API, determines the identity of the caller process and dispatches it to the malice score computer, and takes the appropriate action according to the computation result.
- Authorization Module, which receives information about the `WriteFile` API calls of the process, maintains the table for malice score computation and returns an authorization response according to the predefined threshold.

5.1.1 Intercepting WriteFile API Calls

On Windows platform, controlling file write operations can be achieved in two different ways:

---

[1] Available under GNU v3 at `https://github.com/melparmaksiz/cryptocop`

- user mode, by injecting a Dynamic Link Library (DLL) into target process or modifying the Import Address Table (IAT); or
- kernel mode, by modifying the System Service Dispatch Table (SSDT) or utilizing a file system filter driver.

In our research, we employed DLL injection technique to run our controller code in the target processes. Concordantly, Main Module of CryptoCop is implemented as a DLL. For its technical simplicity, we used AppInit_DLLs method to inject our Main Module into all starting processes. To take control of the program control flow, we utilized Detours library [14] developed by Microsoft Research. Once Main Module is loaded into the target process, it hooks `WriteFile` function, that is, whenever `WriteFile` API is called by a process, the program flow is routed to CryptoCop.

Using AppInit_DLLs Method

The following operations should be performed to utilize `AppInit_DLLs` technique for DLL injection.

- Run Registry Editor Windows Start » Run type regedit.exe ↲
- Navigate to : HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Windows
- Create a string with the name `AppInit_DLLs`, and assign the value `PATH\TO\cryptoCop.dll`
- Create a DWORD with the name `LoadAppInit_DLLs`, and assign the value `00000001`.

After this point, each new process that loads `user32.dll` will also load `cryptoCop.dll`.

5.1.2 Maintaining Malice Score

At each call of `WriteFile` API, CryptoCop acquires the control flow, obtains the location of the file-to-be-written, f. This information is then sent to Authorization Module, which adds f to the trace record of current process p, denoted $Tr(p)$. Each process maintains its own trace record individually by using a hash set such that, at each call, the path of the target file is added here while no two elements are the same.

In our experiments, we assumed that the user has three categories of files:

- *high importance*: files that are critical for the user, i.e., the files for which the user would be willing to pay ransom;

- *low importance*: files that do not contain important data, but losing them would still cause trouble; and
- *dispensable*: files that do not have any value for the user, e.g., temporary files.

Therefore, we defined three possible values for v_f: *high*, *low*, and *none*. Furthermore, we assumed that the user can tolerate the loss of 5 highly important files or 100 low importance files at most. Our model brings the following requirement for the security threshold:

$$5 \times v_f^{high} = 100 \times v_f^{low} \geq T \tag{5.1}$$

The simplest integral solution to Eq 5.1 can be found by computing the least common multiple (LCM) of 5 and 100. Thus, we set

$$T = LCM(5, 100) \tag{5.2}$$

which yields $T = 100$, and accordingly, we have *high*$= 20$ and *low*$= 1$. Finally, we set *none*$= 0$ for dispensable files.

To make the decision of authorization for p, Authorization Module instantiates Eq. 4.1 with $F_p = Tr(p)$, updates M_p and calculates $\Delta_p = M_p - T$. The computation proceeds with checking if $\Delta_p < 0$. If so, then Authorization Module returns ALLOW, the data is written to f and Main Module returns the control to p. Otherwise, i.e., if the process has written some number of files causing the threshold excess, Authorization Module returns DENY and p is terminated.

5.1.3 Malicious Process Termination

Once a process exceeds the security threshold, all subsequent requests of that process are blocked. The process' topmost ancestor is detected. Then, the process tree, of which the topmost ancestor is the root, is terminated in a bottom-up order (i.e from leaves to root).

5.2 Test Environment

This section mentions the steps taken while creating the environment for our experiments for this thesis. We used a Virtual Machine named ransomwareTest with 2 GB memory and dynamically allocated 160 GB hard disk for our tests.

5.2.1 Hardening Windows 7 x86 on VirtualBox

In order to prevent ransomware detecting that it runs in a virtual machine, in which case it doesn't show any malicious activity, the following steps were taken to remove the traces of virtualization from the Virtual Machine (VM).

- **Set Accelaration Type**
  Ransomware checks hypervisor fields in CPUId to look for known VM vendors. Switching `Accelaration Type` from `ParaVirtualization` to `None` helps bypassing this control.

- **Set Processor Count and Disk Size**
  We decided to set processor count to a value greater than one since some ransomware assume single processors as virtual machines as most of the commodity computers today are multi-core. Similarly, since today's computers have large disk capacity, we determined the disk size of 160 GB.

- **Change MAC Address**
  The MAC address of the machine has to be changed, since the default prefix `080027` is a virtual box indicator.

- **Remove VBOX Keywords**
  In order to remove `VBOX` keywords from the output of `sysinfo` console command and Disk properties, we assigned new values to the following keys by using the following command.

  ```
  $ VBoxManage setextradata ransomwareTest key new_value
  ```

  Table 5.1: List of properties with VBOX keyword

Key	Value
VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVendor	LENOVO
VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVersion	LENOVO 1230 Ver 1.00PARTTBL
VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseDate	09/16/13
VBoxInternal/Devices/pcbios/0/Config/DmiSystemVendor	LENOVO
VBoxInternal/Devices/pcbios/0/Config/DmiSystemProduct	Lenovo – 1230 Ver 1.00PARTTBL

- **Antivmdetection Script**
  The AntivmDetection python script developed by Mikael Keri helps to get done additional settings to make VM detection harder such as modification of ProductKey, removing Video Graphics Array (VGA) device.

  It creates two scripts : *.sh and *.ps1. While the file with the sh extension must be run on the host machine, the latter runs on the virtual machine.

```
$ apt-get install python-dmidecode libcdio-utils acpidump mesa-utils
$ wget https://download.sysinternals.com/files/VolumeId.zip
$ wget https://www.nirsoft.net/utils/devmanview.zip
$ git clone https://github.com/nsmfoo/antivmdetection.git
$ echo "meltem" > user.lst
$ echo "meltem-pc" > computer.lst
$ sudo python antivmdetection.py
```

Listing 5.1: Completing the requirements of `antivmdetection.py`.

- **Install Windows 7**

  For our experiments, we decided to work on the X86-32 version of Windows 7 since it is still one of the most common OS.

- **Install Common Applications**

  In order to reflect an authentic user environment, we installed popular applications such as document viewer, office suite, web browser, media player and other system utilities.

  We also cared for special attention to prevent any leakage about the virtualization. In this context, we avoid installing software packages that are developed to improve user experience on guests as they bring virtual device drivers which reveal the virtual environment, e.g., vendor information and known device names.

- **Create Using History**

  In order to thwart advanced fingerprinting of user environment, widely used plugins where appropriate are installed and run the applications for a while to create a usage history.

- **Place Decoys**

  Some decoy files that would be the target of ransomware were generated and placed into the well-known directories such as Desktop, Documents and Pictures.

5.2.2 Making Virtual Machine Vulnerable

In order to further increase the probability of ransomware activity, the following steps were taken to make the VM more vulnerable by turning off the defense mechanisms other than CryptoCop.

- **Disable User Access Control**
  - Open Local Group Policy Editor Windows Start » Run type GPEdit.msc ↵

  - Navigate to : Computer Configuration > Windows Settings > Security Settings > Local Policies > Security Options
  - Set value of:
    - Behavior of the elevation prompt for administrators in Admin Approval Mode to Evaluate Without prompting
    - Detect application installations and prompt for elevation to Disable
    - Run all administrators in Admin Approval Mode to Disable

- **Disable Windows Defender**
  - Open Local Group Policy Editor Windows Start » Run type GPEdit.msc ↲
  - Navigate to : Computer Configuration > Administrative Templates > Windows Components > Windows Defender Antivirus
  - Set value of Turn off Windows Defender to Enable
- **Disable Windows Firewall**
  - Open Control Panel Windows Start » Run type control ↲
  - Navigate to : System and Security > Windows Firewall > Turn Windows Firewall on or off
  - Set value : Turn off Windows Firewall.
- **Disable Windows Updates**
  - Open Control Panel Windows Start » Run type control ↲
  - Navigate to : System and Security > Windows Update > Turn automatic updating on or off
  - Set value : Never check for updates.
- **Deactivate Address Space Layout Randomization (ASLR)**
  ASLR is a memory protection mechanism which prevents buffer-overflow attacks by randomizing the location of system executables in the memory. We deactivated this protection in case this attack might be used by some ransomware.
  - Run Registry Editor Windows Start » Run type regedit.exe ↲
  - Navigate to: HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\Memory Management
  - Create a DWORD with a name MoveImages, and assign the value 00000000.

- **Deactivate Data Execution Prevention (DEP)**
  DEP is a security feature that prevents non-authorized programs to run code from reserved system memory locations. This protection is also disabled to make system more vulnerable by executing the following command in an administrative session of a command prompt.
  `bcdedit.exe /set current nx AlwaysOff`

- **Allow Execution of Powershell Scripts**
  To allow running Powershell scripts, the following command was executed in the power shell with admin rights.
  `Set-ExecutionPolicy Unrestricted`

- **Install Visual C++ Runtime and .NET Framework**
  In order to facilitate the malware samples that leverage Visual C++ runtime and .NET framework, the mentioned software packages were installed.

5.2.3 Network Configuration

In order to work with Cuckoo Sandbox, the VM internet access should be via host-only networking. The following configurations steps were taken to succeed it.

- **Creating a host-only Network**

```
$ vboxmanage hostonlyif create
$ vboxmanage hostonlyif ipconfig vboxnet0 --ip 192.168.100.10
```

LISTING 5.2: Create Host-Only Network

- **Assign Windows Static IP**
  - Open Control Panel Windows Start 》Run type control ↲
  - Navigate to : Network and Sharing Center > Change Adapter Settings > Local Area Connection > Properties > Internet Protocol Version 4
  - Set the values of :
    - IP address to 192.168.100.20
    - Default gateway to 192.168.100.10
    - Preferred DNS server to 8.8.8.8

- **Iptables Rules**

```
$ sudo iptables -A FORWARD  -o wlp1s0  -i vboxnet0 -s 192.168.100.10/24  -m conntrack
 ↪   --ctstate NEW  -j ACCEPT
$ sudo iptables -A FORWARD  -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
$ sudo iptables -A POSTROUTING  -t nat -j MASQUERADE
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Listing 5.3: Iptables Rules For Host-Only Network.

5.3 Cuckoo Sandbox

Cuckoo [5] is an automatic malware analysis system that monitors suspicious files by running them in an isolated environment. We have leveraged Cuckoo to automate our experiments for each ransomware.

5.3.1 VM Snapshots

After creating the test virtual machine described in section 5.2, the Agent file, Cuckoo script that runs inside the Guest and handles the communication and the exchange of data with the Host script, is placed into the VM and run.

We created two different snapshots on this VM for two phases of our experiment :

- Snapshot1: The clean system snapshot to test if a ransomware is still active
- Snapshot2: The hooked system snapshot which CryptoCop is deployed as described in section 5.1.1 without changing any other configuration .

5.3.2 Configuration

The following files stored in the cuckoo configuration directory ($CWD/conf) should be changed.

- In cuckoo.conf, set the values of:
  - machinery to virtualbox
  - ip to 10.0.0.106 (the IP of the host machine on the virtual network interface virbr0)
- In virtualbox.conf, set the values of:

  - under [virtualbox]
    - interface to vboxnet0
    - machines to ransomwareTest
  - under [ransomwareTest]
    - platform to windows
    - ip to 192.168.100.20 (virtual box ip)
    - snapshot to SnapshotX where X $\in \{1, 2\}$

5.3.3 Customization

In order to keep the generated log files separate and secure from being encrypted by ransomware, we customized Cuckoo to create a new directory for each task in a shared directory and to move all related files to this directory. The steps are as follows:

- Processing module should be enabled in the processing.conf (in the cuckoo configuration directory)

```
[cryptocop]
enabled = yes
```

- Processing module should be enabled under the processing label in the common/config.py (in the installation folder)

```
''cryptocop'': {
''enabled'': Boolean(True), },
```

- The script moving files to the shared folder should be placed under the processing directory.

5.3.4 Analysis

Cuckoo starts with the console command : `cuckoo -d`
Cuckoo web starts with the console command : `cuckoo -d web -H 0.0.0.0 -p 8000`

After starting Cuckoo Sandbox, we submitted ransomware samples to analysis for ten minutes from Cuckoo Web Page.

5.4 Experiments

Our experiments can be divided into two phases: (i) building the set of ground truth for active ransomware; (ii) testing CryptoCop against the active ransomware samples.

5.4.1 Building the Ground Truth Set

To evaluate the effectiveness of CryptoCop, we needed to test our prototype implementation against active, real-world ransomware samples – samples that encrypt the user's files. In this respect, we collected malware samples from online repositories, mainly from VirusTotal Threat Intelligence [46] and ViruSign [45]. Next, to obtain the samples which are potentially ransomware, we performed a text-based filtering on malware tags provided by VirusTotal. We used the generic ransomware keywords such as *ransom*, *crypt* and *lock* in the search string and obtained 7132 potential malware samples.

Having the potential ransomware samples at hand, we used Cuckoo automated malware analysis system with the 'Snapshot1' configuration 5.3.1 to filter the active ransomware samples, i.e., collect the malicious executables that performs encryption on user's files.

Once the test system is ready, we initiated the tests by submitting the potential ransomware samples to Cuckoo. In our tests, Cuckoo run each malware sample on the clean snapshot we prepared for this test. While ransomware usually attacks the victim's files immediately after infection, i.e., encrypts the files soon after it is first run, we configured Cuckoo to run each sample at least 10 minutes before concluding the test. After each test, we checked if the decoy files we placed are modified, by comparing the old and new hash values under SHA256, which clearly indicates the existence of a malicious file system activity. If no change occurred during the test, the sample is not count in the ransomware class and excluded from our experiments.

At the end of the first phase of our experiments, we acquired 736 active ransomware samples which formed our ground set.

The reason that the number of active samples being significantly less than the initial set is twofold. First, identification by the AV vendors might not always be accurate. For example, a malware sample that connects to a domain which is also accessed by a ransomware might be mistakenly tagged as "ransomware". Apart from the misidentification, the second reason is that the ransomware may not show any malicious activity at all. There are a wide range of potential reasons that can lead to this result. The ransomware may experience an internal error due to a bug in itself. Also, for a non-autonomous ransomware, the connection to its C&C center may have failed. It may also be possible that our efforts to convince the ransomware that it is running on a real computer might have failed due to its advanced environmental fingerprinting capabilities.

Identifying Ransomware Families

In order to investigate the connections between ransomware samples and to understand the methods used by malware authors, we grouped the tested ransomware samples by their families. To this end, we utilized AVCLASS [37] tool, which automatically labels malware samples using the information provided by AV engines.

For family detection, we used VirusTotal service, which provides scan results that contains the labels assigned by multiple AV vendors. We wrote a Python script, `getVTReports.py`, to download the scan results from VirusTotal. Next, AVCLASS was used to process this information and output the family name based on plurality voting. Listing 5.4 shows the steps of family identification.

```
$ python getVTReports.py --outputDir vtdl
$ python avclass_labeler.py -hash sha256 -vtdir vtdl > labels.txt
$ cat labels.txt
000a77953565d43520dacf7446baef17252718d06fc8770727ee6aacb245db8a   bitman
00b56667d794895fe8342f4d616e303cea222b88d2af8f0042b8e264a759e975   bitman
0a68cf3e31426966aee7d9c76d52df906d7e1fa3380a78e3cde3b56b930f7680   bitman
...
```

LISTING 5.4: Identifying the family names using the VirusTotal reports and AVCLASS tool.

5.4.2 Assessment of CRYPTOCOP Efficiency

After having the ground set ready, we initiated the second phase of the experiment: evaluating the efficiency of the CRYPTOCOP. We tested the active ransomware samples against our CRYPTOCOP prototype by using Cuckoo automated malware analysis system with the 'Snapshot2' configuration 5.3.1. Then we analyzed the CRYPTOCOP reports to assess the detection capability, the number of damaged files and their importance level and the time passed until the detection.

The sample CRYPTOCOP report of a detected Bitman sample can be seen in Appendix A.

5.5 Tools Used In the Experiments

- **Pafish**
  Pafish [31] is a demonstration tool that observes the environment whether it is sandbox and analysis one. It was leveraged to make sure the system is hardened before taking the Snapshot of the test environment.

- **Xenos**
  Xenos [1] is a Windows DLL injector based on Blackbone library. It was used to quickly inject and test our code while developing CryptoCop.

- **Dependency Walker**
  Dependency Walker [28] is a utility tool that examines Windows modules and shows all dependent modules and functions calls from them. It helped to determine which file system function are called by ransomware. Additionally, it was used to make sure if ransomware has dependency on user32.dll, since selected DLL injection method, LoadAppInit_DLL, injects the DLL while being loaded user32.dll.

- **Process Hacker**
  Process Walker [25] is a utility tool that helps monitoring system resources. It was leveraged to discover the ransomware behaviour at runtime, such as which files are being used, disc access volume and frequency, and the processes created by ransomware.

Chapter 6

Results

6.1 Experimental Results

In this section, we report our findings which aim to answer the following research questions:

Q1 Can CryptoCop detect ransomware at an early stage such that the valuable data is protected?

Q2 How CryptoCop affects the activities of benign applications?

Q3 What is the performance overhead of CryptoCop on the host machine on which it acts as an early warning system?

These research questions need to be answered to verify the effectiveness of CryptoCop's design while ensuring its usability. In this regard, **Q1** addresses the security of the system against cryptographic ransomware. As a protection system, CryptoCop must also be usable, since otherwise the users may turn off the defense which makes them vulnerable to ransomware threat which is mostly irreversible. Therefore, we analyze the false positive rate of CryptoCop in **Q2** and its performance impact in **Q3**.

6.1.1 Detection Capability

In this section, we analyze the results of the second phase of our experiments to answer **Q1**, can CryptoCop stop ransomware in an early stage so that valuable data is protected?

We defined the files in Desktop, Downloads, Documents and Picture directories as high protected and set the file loss rate to be at most 5 percent of total files (5 files). We also limited the number of write requests as 100 for low importance files per process.

Table 6.1 demonstrates the detection capability of CryptoCop against real-world ransomware. *WriteFile Statistics* demonstrates the average of the activities of ransomware samples: the number of written files with high importance, the number of written files with low importance, and total execution time until detection, i.e., $\Delta_p \geq 0$. During the experiments, CryptoCop stopped 706 out of 736 ransomware samples once they reached the security threshold T. We left the discussion on the undetected samples to §7. The results prove that our approach of detecting ransomware activity only by monitoring file write API is an effective measure.

6.1.2 Usability Tests

In this section, we first answer **Q2**, how CryptoCop interprets the legitimate activities on the system by testing benign applications and observing the false-positive rate of CryptoCop. We run our experiments on a X86-32 version of Windows 7 virtual machine, with 2GB of RAM and 2 CPU cores at 1.90GHz. We set the importance value of files in temp folder and temporary internet files to *none*, since there is no user specific data in these folders.

Analysis of False Positives In the course of answering **Q2**, to measure false-positive rate, we ran Windows common applications: Chrome, 7-Zip, Dropbox, Acrobat Reader, Sticky Notes, Internet Explorer, Skype, Telegram Desktop, µTorrent, MS Word, MS Excel, GIMP, LibreOffice. We did not notice slowdown in any of the applications.

In our experiments on the hooked system, we detected that 7-Zip and µTorrent might exceed the threshold, e.g., when multiple files extracted by 7-Zip or downloaded by µTorrent. LibreOffice creates a temporary file on each save operation such that excessive number of saves causes a false positive, especially when working on a high importance file. With the rest of the applications, we performed the following tests and no false positives occurred. We browsed the Internet with Chrome for two hours and downloaded & installed 7-Zip and Dropbox. We installed Dropbox, logged in and synchronized around 3000 files with cloud. We created 3 files and updated a couple of files without any problems. We logged in a Skype account, sent/received files, and made a call. Table 6.2 demonstrates the evaluation of CryptoCop's usability against common benign applications in terms of performed operations and number of modified files.

It should be noted that the burden caused by the false positive scenarios can be mitigated by asking for user approval upon suspicious activity detection. If the user decides that the process is in fact a legitimate application, then it can continue to run.

Table 6.1: Evaluation of CryptoCop against real-world & active ransomware samples.

Family	Number of Samples (stopped / total)	`WriteFile` Statistics		
		Avg. High Importance	Avg. Low Importance	Time (ms)
Barsys	1/1	1	80	5.74
Bitman	147/149	1.18	76.46	131.13
Cerber	58/58	4	25.74	125.42
Cryakl	1/1	0	100	8.77
Crypmod	3/4	0.67	86.67	3606.04
Cryptowall	0/1	-	-	-
Cryptxxx	2/2	2.5	53	156.33
Crysis	3/3	0	100	1839.53
Dalexis	0/3	-	-	-
Deshacop	1/1	1	80	3.14
Enestaller	1/1	4	25	90.20
Enestedel	1/1	4	25	6673.11
Gamarue	2/2	1.5	70	1614.51
Gandcrab	1/1	0	100	2.64
Jaff	1/1	0	100	7.7
Lethic	4/4	1	80	4.30
Locky	42/45	3.90	26.52	366.34
Midie	1/1	1	80	708.97
Mikey	1/1	1	80	2.44
Neoreklami	1/1	0	100	11.7
Petya	2/2	5	2	16.5
Razy	6/6	2.5	52.17	4.12
Saturn	0/1	-	-	-
Scar	3/3	1.33	73.33	6.79
Scatter	2/2	4	23.5	3761.35
Shade	2/2	2.5	58	1922.88
Shiz	16/17	1.25	75	8.70
Spora	2/2	4.5	16.5	257.84
Tescrypt	4/5	1.25	75	6.27
TeslaCrypt	306/309	1.24	75.23	61.93
Tpyn	1/1	1	80	3.45
Upatre	0/7	-	-	-
Virlock	0/3	-	-	-
WannaCry	0/1	-	-	-
wowlik	1/1	5	4	10802.45
Wyhymyz	1/1	0	100	1207.86
Yakes	34/37	4.85	70.94	13.62
Zerber	49/49	4	25	172.76
Zusy	6/6	1.17	76.67	5540.31
Total:	706/736 (95.9 %)			

TABLE 6.2: `WriteFile` API call statistics of common applications and their interactions with CRYPTOCOP. *Operation* represents the functionality of the application. *WriteFile Statistics* demonstrates the number of `WriteFile` API calls during our tests: *High Protected* column displays the number of write operations on high importance files, and *Low Protected* for those with low importance. The red lines show the program activities that might exceed the threshold for high or low protected files.

Application	Operation	`WriteFile` Statistics	
		High Importance	Low Importance
Acrobat Reader	File Open	0	1
Chrome	Browsing	0	1
	Download	0	0
Dropbox	Install	1	1
	Log In	0	3
	Synchronization	0	0
	Add File	0	0
GIMP	Open & Edit File	1	0
Internet Explorer	Browsing	0	1
	Download	1	0
LibreOffice	Open a File	1	0
	Edit a File	3	0
	Ctrl+S	1	0
	Multiple Ctrl+S	5	100
Microsoft Excel	Open & Edit File	0	0
Microsoft Word	Open & Edit File	0	0
Skype	Log in	0	1
	Sent File	0	0
	Received&Save File	0	0
	Call	0	0
Sticky Notes	Installation	1	2
	Take Notes	0	0
uTorrent	Download one file	0	1
	Download multiple files	5	100
WinRAR	Compress	1	0
	Extract one file	1	0
	Extract multiple files	5	100
7zip	Compress	1	0
	Extract one file	0	1
	Extract multiple files	5	100

6.1.3 Performance Overhead

In this section, we look for the answer of **Q3**, determine the performance impact of CRYPTOCOP on the host system by measuring the overhead on calls to file write API. We run our experiments on a x86 version of Windows 7 virtual machine, with 2GB of RAM and 2 CPU cores at 1.90GHz.

In the course of answering **Q3**, we called `WriteFile` API to overwrite 1 KB, 32 KB, 1 MB and 4 MB of data inside a file which is randomly selected from a set of 100 files and measured the time taken during these operations. We limited the number of different files to a constant number to stay below the threshold limits of CryptoCop. We performed 1 million iterations of this experiment, which was observed to be sufficient enough for average time to converge/stabilize on both clean and hooked systems. CryptoCop caused the average write time to increase around `0.1ms` per call.

Table 6.3: Benchmark Results

Platform	Length of Written Data			
	1 KB	32 KB	1 MB	4 MB
Clean System	109 μs	115 μs	295 μs	4443 μs
CryptoCop	238 μs	247 μs	408 μs	4526 μs
Overhead	129 μs	132 μs	113 μs	83 μs
Relative Overhead	54 %	53 %	27 %	0.1 %

As Table 6.3 demonstrates, the time spent for CryptoCop controls is independent of the size of the buffer. Additionally, the slowdown in the write performance is due the fact that the time consumed by `WriteFile` is exceedingly short for small chunk of data and becomes comparable to time taken by CryptoCop. It can be seen that as the size of the data increases, the impact of the control becomes negligible.

We also remark that the benchmarks are performed using the prototype developed during this research and no optimization was made. It is reasonable to expect better performance results from an optimized version of CryptoCop.

Chapter 7

Discussion

In this chapter, first, we compare the state-of-the-art anti-ransomware systems in the cryptographic literature, including our system, CryptoCop. Next, we point out the potential limitations of current prototype, explain the design challenges, and discuss how CryptoCop can address them in the future.

7.1 Comparison with the Other Anti-Ransomware Systems

In Table 7.1, We present the outstanding defense solutions that were described in Section 3. *Detection Approach* denotes the operation mode of the corresponding anti-ransomware, according to the categorization given in Section 3. Note that, RWGuard is labeled as Hybrid as it unifies various approaches such as behavioral analysis, API hooking and decoy files.

The anti-ransomware systems can be classified into three groups according to their security function; prevention, detection, and recovery. CryptoDrop, which is an example of the *detection* group, looks for the signs of ransomware activity by monitoring the running processes, that detection occurs *during* the attack, in run-time. On the other hand, UShallNotPass is an example of *prevention* group: it stops non-whitelisted applications that access to CSPRNG APIs *before* any damage occurs to any file. In contrast, PayBreak attempts to recover files *after* the attack which makes it a member of the *recovery* group.

Based on their defense strategy, some of the anti-ransomware systems are sensitive to *obfuscation*. For instance, PayBreak is designed to hook only known APIs. Consequently, it cannot hook the cryptographic function calls if the code is obfuscated. Similarly, CfHk module of RWGuard can only hook system-provided crypto APIs, therefore cannot recognize functions from third-party crypto libraries. Other anti-ransomware systems are not sensitive to obfuscation, according to our research.

Lastly, except CryptoCop, all behavioral detection systems look for anomalies in disk I/O, such as entropy increase or file type changes, and therefore can be tricked by the next generation ransomware as we mentioned in Section 4.1. On the other hand, CryptoCop does not rely on the statistical analysis of the write buffers which enables it to be I/O oblivious. Likewise, hardware-assisted solution FlashGuard, and network-level defense systems [3] and [6] also works independently of the disk I/O.

Table 7.1: Comparison of Ransomware Defense Systems

System	Defense Approach	Protection Mechanism	Obfuscation Agnostic	I/O Oblivious
UNVEIL	Behavioral Analysis	Detection	✓	✗
CryptoDrop	Behavioral Analysis	Detection	✓	✗
ShieldFS	Behavioral Analysis	Detection	✓	✗
PayBreak	Key-escrow	Recovery	✗	✓
Redemption	Behavioral Analysis	Detection	✓	✗
UShallNotPass	Access Control	Prevention	✓	✓
DaD	Behavioral Analysis	Detection	✓	✗
RWGuard	Hybrid	Detection	✗	✗
FlashGuard	Hardware Assisted	Recovery	✓	✓
Cabaj et al. [3]	Network Level	Detection	✓	✓
Cusack et al. [6]	Network Level	Detection	✓	✓
CryptoCop	Behavioral Analysis	Detection	✓	✓

7.2 Limitations

The history of the combat with malware suggests that the ransomware mitigation is an arms race that never ends. It is possible that the ransomware authors will develop new samples that use more advanced techniques than now. In this section, we will point out and discuss the limitations of CryptoCop.

CryptoCop is designed as an early-warning system to detect cryptographic ransomware. This design decision brings up both advantages and disadvantages. On one side, CryptoCop is highly efficient in detecting ransomware with a minimal performance overhead. This makes CryptoCop a suitable defense mechanism in real-world systems. On the other hand, while our system minimizes the damage of ransomware, it does not guarantee zero data loss. However, it should be noted that even commercial anti-ransomware provide a similar protection level.

The proposed solution itself is independent of operating system; however our prototype and tests are performed on Windows 7 as malware commonly targets Windows operating system [2, 43] and the market share of Windows 7 is still one of the highest [39].

Our prototype CryptoCop, as described in Section 5.1, utilizes user-mode hooks to monitor and control another user-mode `WriteFile` API. While this approach is useful to give an easy to implement proof-of-concept, there are two issues to be addressed. First, ransomware may prevent our control by attacking to our hooks first. This is a limitation of user-mode hooking. Secondly, the ransomware may use low-level file write functions, e.g., WannaCry, that CryptoCop is not set up to control, to evade detection. This is an accepted risk that we take when implementing our research level prototype. Namely, our prototype is designed to operate in user mode as we explained in Section 5.1, therefore limited to capture only the documented user mode APIs. In particular, since `NtWriteFile` is an undocumented, low-level Windows API, our prototype cannot intercept or control `NtWriteFile` calls. However, both of these issues are purely related to the implementation choices and do not invalidate our design principles. We will develop a file system mini-filter driver to overcome these limitations in the future.

It can be possible that ransomware can create temporary files and overwrite protected files via move operation, however, we decided to leave out additional controls in order to avoid processing overhead as we didn't observe this scenario in our experiments on 736 samples. Furthermore, this family will still be detected due to write activity in non-protected folders but with increased latency.

To evade detection, ransomware can create new processes for encrypting a subset of files which would allow it to remain undetected as the Malice Score of each process would be below the security threshold. Moreover, the ransomware might fork processes with a different parent. As a result, they can not be tracked/terminated as members of the same process tree. Detection of this attack strategy requires interception of process create calls and complex bookkeeping of malice scores which is left as a future work.

The ransomware might also have advanced attack capabilities such as NotPetya which can reboot the victim's machine and load its own kernel to perform disk-level encryption. In this case, the host OS would not be up and hence CryptoCop will not be active. Therefore, such ransomware would be able to bypass the protection offered by CryptoCop. However, OS vendors recently started to take cautions to reduce this attack surface. For example, Secure Boot allows loading a kernel only if it comes from a trusted source by utilizing digital signatures. However, this technology is available only on supported hardware and OS.

Recently, with the hope of encrypting valuable data, new ransomware variants started targeting database servers. These new variants might (i) attempt to authenticate to a database and encrypt all the records; (ii) attack to vulnerable web APIs and perform SQL Injection to execute encryption commands; and (iii) encrypt the database files on the victim machine. CryptoCop will not be able to detect SQL injection attacks since they happen inside the database process, however, it can detect the direct attacks to database's files. The detection capability for this scenario heavily depends on the implementation details of the storage engine, the success rate will increase if the database maintains separate files per tables. Furthermore, following the best practices in database security might prevent the vast majority of ransomware attacks targeting databases. For instance, securing database authentication, e.g., choosing a secure password would prevent ransomware from accessing the database. Penetration tests would help system administrator fix the vulnerable APIs and prevent ransomware attacks from that surface. Likewise, managing access control to database files and processes, i.e., running database process under a separate user account [1] and granting database files' write access only to that user, would help mitigate ransomware targeting database files.

Instead of trying to encrypt the files on victim's computer, ransomware might attempt to infiltrate the important data, e.g., blueprints of a patent and trade secrets, prioritize them in reasonable amounts and transfer to a remote server. To prevent victim's access, the local copies must be deleted or overwritten. In case of deletion, CryptoCop needs to be adjusted to listen delete file API calls, however we believe ransomware would rather overwrite the data since deleted files can easily be recovered by recovery tools. In the latter case, monitoring write operations would still be an effective measure. Additionally, as the nature of this attack would involve network traffic, the defense mechanism can be enhanced with specialized tools to sniff and inspect the traffic. Developing an augmented system which works side-by-side with CryptoCop would be useful to mitigate such a new ransomware variant. Still, it should be noted that, once the important data is stolen, there would be no guarantee to prevent cyber-criminals from demanding multiple ransoms via blackmailing if the data is highly confidential/sensitive.

As an enhancement, upon ransomware detection, CryptoCop can dump the process memory and extract the keys used by ransomware in order to recover the encrypted data. However, this would not guarantee recovery of all files because old keys might have already been erased in cases where ransomware uses different keys for each file.

Finally, to increase the usability of CryptoCop, we concluded four improvements: (i) asking users approval for termination of the suspicious process, (ii) ability to identify

---

[1] Making the owner of the database process a separate account might prevent ransomware from terminating that process. Consequently, the ransomware will not be able to delete the database file since the database process would be active.

trusted applications, (iii) handling Malicious Score, calculated by written unique files and their importance level, as a cache (with a suitable eviction algorithm) to avoid long-time processes exceeding the threshold over time, and (iv) excluding processes that performs only write operations without reading files which are all left for future work for the sake of simplicity. Additionally, the usability of CryptoCop should be tested under various workloads. For different user-profiles, Malice Score Computation should be switchable to different configurations. We leave the task of researching multi-user environments for future work.

Chapter 8

Conclusion

This thesis presents a novel approach to detect and halt ransomware with low cost. The approach leverages the basic and inevitable operation of ransomware: write requests to file system. Controlling `WriteFile` API calls gives zero chance to Ransomware to modify files more than the defined threshold.

We designed and implemented a prototype of the approach, called CryptoCop. It mainly has two components: (i) Main Component intercepting `WriteFile` API calls and controlling the flow and (ii) Malice Score Computation calculating the malicious score of process.

The carefully engineered virtual test environment prevents Ransomware to detect that it is running on a virtual environment since Ransomware samples might stay inactive on a virtual environment to avoid being tested. Additionally, the experiments are aimed to be performed under as realistic conditions as possible.

In order to measure the efficiency of approach, we collected real-world Ransomware samples on the web and weed out inactive ones by leveraging the prepared test environment. Then, the active ransomware samples were run in the test environment with the help of Cuckoo.

In our experiments, CryptoCop successfully detected 706 out of 736 samples since it is oblivious to the techniques used by families but only relies on the write calls. The reason for missing some ransomware samples can be explained by implementation deficiencies: hooking technique and low level file write functions. Additionally, the result of our usability experiments show that the approach is ready for users with some additional utilities.

Since the approach works on the context of a single process and intercepts only single function call, no additional synchronization and communication costs are added, it adds a minimal effect to the system. For these reasons, our approach is effective, suitable for real-world and ready for any new versions of ransomware.

To sum up, this work proposes a solution based on the analysis of file write activity considering the fact that file write is a vital operation of ransomware. Our prototype is proven to successfully detect 96.5% of active ransomware samples. Obviously, to achieve robust and user-friendly experience in wide scale, the prototype needs more detailed work in system integration and usability. These are left as future work.

Appendix A

Report of a Bitman Sample

```
1552118376437 [ATTACHED]
1552118382715 [HIGH]  [MALICIOUSSCORE] 20 C:\Users\meltem\Documents\recover_file_skkwbbmcc.txt
1552118382840 [LOW]  [MALICIOUSSCORE] 21 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$IIOTCDF.pdf
1552118382871 [LOW]  [MALICIOUSSCORE] 22 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$IJYDCL4.pdf
1552118382887 [LOW]  [MALICIOUSSCORE] 23 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$IRBQ7F1.pdf
1552118382903 [LOW]  [MALICIOUSSCORE] 24 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$IU9LOXK.pdf
1552118382918 [LOW]  [MALICIOUSSCORE] 25 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$RIOTCDF.pdf
1552118382950 [LOW]  [MALICIOUSSCORE] 26 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$RJYDCL4.pdf
1552118383168 [LOW]  [MALICIOUSSCORE] 27 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$RRBQ7F1.pdf
1552118383528 [LOW]  [MALICIOUSSCORE] 28 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\$RU9LOXK.pdf
1552118383871 [LOW]  [MALICIOUSSCORE] 29 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\help_recover_instructions+rkd.png
1552118383918 [LOW]  [MALICIOUSSCORE] 30 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\help_recover_instructions+rkd.txt
1552118383934 [LOW]  [MALICIOUSSCORE] 31 C:\$Recycle.Bin\S-1-5-21-786200809-895886964-1289355992-1000\help_recover_instructions+rkd.html
1552118383981 [LOW]  [MALICIOUSSCORE] 32 C:\$Recycle.Bin\help_recover_instructions+rkd.png
1552118384075 [LOW]  [MALICIOUSSCORE] 33 C:\$Recycle.Bin\help_recover_instructions+rkd.txt
1552118384075 [LOW]  [MALICIOUSSCORE] 34 C:\$Recycle.Bin\help_recover_instructions+rkd.html
1552118384153 [LOW]  [MALICIOUSSCORE] 35 C:\Users\help_recover_instructions+rkd.png
1552118384184 [LOW]  [MALICIOUSSCORE] 36 C:\Users\help_recover_instructions+rkd.txt
1552118384200 [LOW]  [MALICIOUSSCORE] 37 C:\Users\help_recover_instructions+rkd.html
1552118384200 [LOW]  [MALICIOUSSCORE] 38 C:\eula.1028.txt
1552118384231 [LOW]  [MALICIOUSSCORE] 39 C:\eula.1031.txt
1552118384262 [LOW]  [MALICIOUSSCORE] 40 C:\eula.1033.txt
1552118384262 [LOW]  [MALICIOUSSCORE] 41 C:\eula.1036.txt
1552118384285 [LOW]  [MALICIOUSSCORE] 42 C:\eula.1040.txt
1552118384305 [LOW]  [MALICIOUSSCORE] 43 C:\eula.1041.txt
1552118384305 [LOW]  [MALICIOUSSCORE] 44 C:\eula.1042.txt
1552118384321 [LOW]  [MALICIOUSSCORE] 45 C:\eula.2052.txt
1552118384321 [LOW]  [MALICIOUSSCORE] 46 C:\eula.3082.txt
1552118384383 [LOW]  [MALICIOUSSCORE] 47 C:\PerfLogs\Admin\help_recover_instructions+rkd.png
1552118384399 [LOW]  [MALICIOUSSCORE] 48 C:\PerfLogs\Admin\help_recover_instructions+rkd.txt
1552118384399 [LOW]  [MALICIOUSSCORE] 49 C:\PerfLogs\Admin\help_recover_instructions+rkd.html
1552118384461 [LOW]  [MALICIOUSSCORE] 50 C:\PerfLogs\help_recover_instructions+rkd.png
1552118384493 [LOW]  [MALICIOUSSCORE] 51 C:\PerfLogs\help_recover_instructions+rkd.txt
1552118384493 [LOW]  [MALICIOUSSCORE] 52 C:\PerfLogs\help_recover_instructions+rkd.html
1552118384539 [LOW]  [MALICIOUSSCORE] 53 C:\Python27\DLLs\help_recover_instructions+rkd.png
1552118384571 [LOW]  [MALICIOUSSCORE] 54 C:\Python27\DLLs\help_recover_instructions+rkd.txt
1552118384571 [LOW]  [MALICIOUSSCORE] 55 C:\Python27\DLLs\help_recover_instructions+rkd.html
1552118384633 [LOW]  [MALICIOUSSCORE] 56 C:\Python27\Doc\help_recover_instructions+rkd.png
1552118384664 [LOW]  [MALICIOUSSCORE] 57 C:\Python27\Doc\help_recover_instructions+rkd.txt
1552118384664 [LOW]  [MALICIOUSSCORE] 58 C:\Python27\Doc\help_recover_instructions+rkd.html
1552118384727 [LOW]  [MALICIOUSSCORE] 59 C:\Python27\include\help_recover_instructions+rkd.png
1552118384743 [LOW]  [MALICIOUSSCORE] 60 C:\Python27\include\help_recover_instructions+rkd.txt
1552118384758 [LOW]  [MALICIOUSSCORE] 61 C:\Python27\include\help_recover_instructions+rkd.html
1552118384758 [LOW]  [MALICIOUSSCORE] 62 C:\Python27\Lib\abc.py
1552118384805 [LOW]  [MALICIOUSSCORE] 63 C:\Python27\Lib\aifc.py
1552118384868 [LOW]  [MALICIOUSSCORE] 64 C:\Python27\Lib\antigravity.py
1552118384868 [LOW]  [MALICIOUSSCORE] 65 C:\Python27\Lib\anydbm.py
1552118384883 [LOW]  [MALICIOUSSCORE] 66 C:\Python27\Lib\argparse.py
1552118384914 [LOW]  [MALICIOUSSCORE] 67 C:\Python27\Lib\ast.py
1552118384946 [LOW]  [MALICIOUSSCORE] 68 C:\Python27\Lib\asynchat.py
1552118384977 [LOW]  [MALICIOUSSCORE] 69 C:\Python27\Lib\asyncore.py
```

```
1552118384993 [LOW]  [MALICIOUSSCORE] 70 C:\Python27\Lib\atexit.py
1552118385008 [LOW]  [MALICIOUSSCORE] 71 C:\Python27\Lib\audiodev.py
1552118385039 [LOW]  [MALICIOUSSCORE] 72 C:\Python27\Lib\base64.py
1552118385071 [LOW]  [MALICIOUSSCORE] 73 C:\Python27\Lib\BaseHTTPServer.py
1552118385071 [LOW]  [MALICIOUSSCORE] 74 C:\Python27\Lib\Bastion.py
1552118385086 [LOW]  [MALICIOUSSCORE] 75 C:\Python27\Lib\bdb.py
1552118385102 [LOW]  [MALICIOUSSCORE] 76 C:\Python27\Lib\binhex.py
1552118385133 [LOW]  [MALICIOUSSCORE] 77 C:\Python27\Lib\bisect.py
1552118385149 [LOW]  [MALICIOUSSCORE] 78 C:\Python27\Lib\bsddb\db.py
1552118385164 [LOW]  [MALICIOUSSCORE] 79 C:\Python27\Lib\bsddb\dbobj.py
1552118385180 [LOW]  [MALICIOUSSCORE] 80 C:\Python27\Lib\bsddb\dbrecio.py
1552118385180 [LOW]  [MALICIOUSSCORE] 81 C:\Python27\Lib\bsddb\dbshelve.py
1552118385211 [LOW]  [MALICIOUSSCORE] 82 C:\Python27\Lib\bsddb\dbtables.py
1552118385227 [LOW]  [MALICIOUSSCORE] 83 C:\Python27\Lib\bsddb\dbutils.py
1552118385227 [LOW]  [MALICIOUSSCORE] 84 C:\Python27\Lib\bsddb\test\test_all.py
1552118385243 [LOW]  [MALICIOUSSCORE] 85 C:\Python27\Lib\bsddb\test\test_associate.py
1552118385258 [LOW]  [MALICIOUSSCORE] 86 C:\Python27\Lib\bsddb\test\test_basics.py
1552118385274 [LOW]  [MALICIOUSSCORE] 87 C:\Python27\Lib\bsddb\test\test_compare.py
1552118385321 [LOW]  [MALICIOUSSCORE] 88 C:\Python27\Lib\bsddb\test\test_compat.py
1552118385336 [LOW]  [MALICIOUSSCORE] 89 C:\Python27\Lib\bsddb\test\test_cursor_pget_bug.py
1552118385352 [LOW]  [MALICIOUSSCORE] 90 C:\Python27\Lib\bsddb\test\test_db.py
1552118385352 [LOW]  [MALICIOUSSCORE] 91 C:\Python27\Lib\bsddb\test\test_dbenv.py
1552118385368 [LOW]  [MALICIOUSSCORE] 92 C:\Python27\Lib\bsddb\test\test_dbobj.py
1552118385368 [LOW]  [MALICIOUSSCORE] 93 C:\Python27\Lib\bsddb\test\test_dbshelve.py
1552118385399 [LOW]  [MALICIOUSSCORE] 94 C:\Python27\Lib\bsddb\test\test_dbtables.py
1552118385414 [LOW]  [MALICIOUSSCORE] 95 C:\Python27\Lib\bsddb\test\test_distributed_transactions.py
1552118385414 [LOW]  [MALICIOUSSCORE] 96 C:\Python27\Lib\bsddb\test\test_early_close.py
1552118385430 [LOW]  [MALICIOUSSCORE] 97 C:\Python27\Lib\bsddb\test\test_fileid.py
1552118385446 [LOW]  [MALICIOUSSCORE] 98 C:\Python27\Lib\bsddb\test\test_get_none.py
1552118385446 [LOW]  [MALICIOUSSCORE] 99 C:\Python27\Lib\bsddb\test\test_join.py
1552118385461 [LOW]  [MALICIOUSSCORE] 100 C:\Python27\Lib\bsddb\test\test_lock.py
1552118385461 [MAXWRITE_EXCEEDED]
1552118385461 [EXIT_PROCESS]
1552118385461 [PARENT PROCESS SEARCH] nthfrgd.exe:3176
1552118385461 Parent: 2616 Child: 3176
1552118385461 2616 Open Process Failed. Return empty as name. Error Code:87
1552118385461 [PARENT PROCESS SEARCH] :2616
1552118385461 Parent not exist  : 2616
1552118385477 2616 Open Process Failed. Return empty as name. Error Code:87
1552118385477 [KILL PROCESS TREE] :2616  KILLER PROCESS:3176
1552118385477 [KILL PROCESS TREE] nthfrgd.exe:3176  KILLER PROCESS:3176
1552118385477 [KILL PROCESS] 2616
1552118385477 [KILL MAIN]
```

Appendix B

Active Ransomware Samples

Table B.1 lists the active ransomware samples which forms the ground truth set used in the experiments. Family names are determined by AVClass tool [37], as we described in Section 5.4.1.

Table B.1: Ground Truth Data Set.

#	SHA256 Digest	Family
1.	afe7d0ce397a44c9740cc1a8e3434bb2858d4ab1f5ee85ad17d61952e72e58bc	barys
2.	000a77953565d43520dacf7446baef17252718d06fc8770727ee6aacb245db8a	bitman
3.	00b56667d794895fe8342f4d616e303cea222b88d2af8f0042b8e264a759e975	bitman
4.	0a68cf3e31426966aee7d9c76d52df906d7e1fa3380a78e3cde3b56b930f7680	bitman
5.	0af44523884f1b6c6d033fbb13fe7383a007219b57b1a1afc9bd44b84b4d6d52	bitman
6.	0afd142d30bb173793438ccfd427624532cc681b27806f0b5e1c3d8777aeaf97	bitman
7.	0b89cae095d2375b6781a33ef28a858e26cb13e4545e2f4e017803d9efa7648c	bitman
8.	0ba325cbaeefed253e6dc0ff2b7ae4e578f8f7b348f4e451499323104f9b3d69	bitman
9.	0bf5a007fe8c1a4f68b16d99febb5d30f4562cb175cdc4ae7747fc5d6b3cc36d	bitman
10.	0c7d0ad0454b780997f5d6856cf87920e9fe667634bd8ccab53177d5f32de5a2	bitman
11.	0cd8fa6de1e03962d36497690edda81c6a9e033c0f7e93121af60f2aef0c0be9	bitman
12.	0ce6dfaf77e59c2a01d4aa15ce4387d65e968a97ea6322c573e5b04e204bd293	bitman
13.	0d38539d70a1957602d9a6e40bd30702c724d648a4dba5fe985027a38ad7c4cf	bitman
14.	0d82b1678d9a7b935c375c3a76e26e77c8d78751ed7db72d21be6bb790d8d1d0	bitman
15.	0db239807f0de42d485dd30ac02de02dd844b6984688aa65ba578c7cd9f02b96	bitman
16.	0dc773242fe5ccb38e382b52f2182d9833e3cc4d299dd4a5ff2c83e054bc82ac	bitman
17.	0e8e6a467e2f65f767a4d5a811bcafd4db0a6aee0436d19c7cda9f9184d7dcab	bitman
18.	0edbf805708a74d4eb4b1d53d044a48825a1ba4eba1d1334d57c6fb3cd3969b3	bitman
19.	0ee0da5dd380639c256b0cc97a7b7773d8a5881d0b30608f73fcd81cf6b271c6	bitman
20.	0f03877e986a07a5f8d5340e45c499e72226707876076fc8e6dddc1d26e21094	bitman
21.	0f5a259f8bd9ae4be2c73894a287cbac9db737b182ee9e694282aaf1fa9af1fe	bitman
22.	0fc7f6584fcfd7af6b7296ed32d006429f777b6673c654172a0c84ba2111950a	bitman
23.	0fd0a87376a2ddd4709d53d9f91dcb8ddddb36b14e9fe688093455ed7f240280	bitman
24.	0fdb9c04debec09ed53f34accfba6e087e9dbe00279a3ed92004e90649d9def5	bitman
25.	a027e30ff931dd1235d991ee1ae72d72b3e7e78c9f708920b31c1b9fbb9dbf11	bitman
26.	a07b496d7a40ebafd3dbe348305d363eaa60e1ba17e67888969b458b4feef565	bitman
27.	a098e8ba6601472a2b3e5f25c397e06be410c7db6e39ad14766eae09a9322c77	bitman
28.	a0acbd92cd619324daa2121c15c3291bdee7dd785f9f09d0fdcb6ace49072f22	bitman
29.	a0b21558804bf306650fe6a44104423fd55488585b99703af47530723b64f3af	bitman
30.	a0b9b6cfafe5dd2f124d5cb345a2198c8739fa18b733e1044f0c38c5d15a5390	bitman
31.	a0fa488fac8ff118dee224fb4c6a7d5fce7c5ddfcfcd20f736fe91289ae408a0	bitman
32.	a10886568925ae74c6afc92e9dba804d3f48692445eb0c022c1888dcc9e8ee26	bitman
33.	a1c5387082247bd24188b0e3912e55a766e573d5651502022543f7292278963b	bitman
34.	a1ca8bed01129c81c7b383cd5b1aea7d907ffd6e5c3337e7bbad2b613f921966	bitman
35.	a1fb8fde3db83d201fe65bbd8c2e046cb9e03e5a7b791a071529a8e8d7213e8d	bitman
36.	a20a0f9939017a46c5c99cf38143c009cbef5685f0ac8d5063bb709e1ffed7d1	bitman
37.	a2569ded9fecf87eaeebff6da06b9ad723a23979e8e6c233a13cc5e2ef6f7270	bitman
38.	a2877426e8a8910e386d99a37b9f390cd0192ff3a48bd205bae9043127874211	bitman

39.	a2b2c46b99d8f6eeb1478c7df8ffcd89829f3bd166ac2c533857e53b0075e05f	bitman
40.	a34da806daf10d4ea434062371c4dca4ad91278440971baec3453bc4d05603d6	bitman
41.	a34eb51fc9cc40a34d89e9de8a5fa7df039643cde5a5c8eff5ece6afddc0f102	bitman
42.	a3baf5ad1f16fab147d8fe60b78af535fa7c3b95debf20c0fc9b77f4d4e4e874	bitman
43.	a3fecd9da4eb3e4b8c35fff66b4771888580900aaad2ba6e56b4fb174f83f44c	bitman
44.	a4c6f8eb783947ab52e63ee5661112ec9c0d4392b41963768c146573d025c821	bitman
45.	a4cad8d7a5e3fc1bf31d49e9b2dd9a358326acb0902a1f802833d28553f50d80	bitman
46.	a4f85cb268d38c519e5a880bbc3df0552e80f436041b0a178affd6c3f4b8496c	bitman
47.	a509653a92f1e4ab08d72e0a3b98120eadc5838a35394460b42bbc75681ab4bd	bitman
48.	a5337ab1d08cfcc0ae17848a9abd4eadf0a9ada9bb7bb894bb903e3917c74110	bitman
49.	a5fbf7335effd03a454fa0e68728fde49df05d6c6d1bbc4c7875d43e26a0e1b6	bitman
50.	a7128843746054df3c00aa31b5285df50dd4c9f7f410a686b020d2f2ea22dfcd	bitman
51.	a7f52e451f6ea384a2cd532cf6b75aa85381e0eab9989f29dc0e9ac7c576d2e4	bitman
52.	a80c0e59be20c7154ad3007c5e1653c54e76393439779f5644b05d5b49ff2cf4	bitman
53.	a84438b74d5cfcad7007d845d42d905c9a43df0d0ad45d23bcf05d2ad8e39039	bitman
54.	a89905505033361021d0def1c11b087c50d7c08633559b4785a54c05f5a884e3	bitman
55.	a8e6c87180a0a26dc5e67c02a582625c93331a8623b76a8b948a09a57f181522	bitman
56.	a9c5f80e925f52e238ef80b6eec5f3f340ba52f2bb89d179bff0533dfcb4f2a8	bitman
57.	a9f98a23d95ae13c9a5df11621da6b2733c3f4283159230347d081588b4e3bbb	bitman
58.	aa96defdc0aae52628fccc5f77ca62a747dc5e8cfe1fa6bc8f4f530301fd7017	bitman
59.	aa9af375460f7ebfc7fc02a69b6f3958687675b34bd931a6f0e34ba7b1550100	bitman
60.	aac12810ab485ee5cd3ca0b20400a6e96406c61727c98a0f9c26f5d583a37b31	bitman
61.	aadd5efb71bb89bb34a1e9c5b74263650db58e86282500274ec60483471ee616	bitman
62.	ab7edf1703a309e88b103cf7b70b7169be8e0e1f1c60551f51faab883b67148e	bitman
63.	abdb3e8b6723fa7fc12273163154303ae33aab9bb734caf8707f982b63bee34f	bitman
64.	abe845169754443d13dc3583de98525cdb9d1250377a3187d4fe21eb10547055	bitman
65.	ac0337476d12e1a688773a4190c62533f8854d28359d1204ccd732a7c92420c9	bitman
66.	ac354dc0809277e06a6578abaa95337e5dbc4b92ec48faa27aecae2ff52601b4	bitman
67.	ac42cecb88ab88b7c32caf954ce9586900a32b57f14be539bb5aaef71762619d	bitman
68.	ac58a1eabe7692566d8db14d5a92814b7e15fea2d94f305300f356ad92b9bc4d	bitman
69.	acb67c79c1dedf069848affbbf9f7fb1074901d1ba47d3a761407048a483ac93	bitman
70.	ad32d7198db0104b0998ce4dd7d8e73e3d8450584bcc972214ffd499785e9df7	bitman
71.	adf41357456704f4620b5a6d422927fb24c41a9b79fe73aedc201f82e81f0ffd	bitman
72.	adf937cc743068392d1b935b1a2c5cdea1283de81946d9b291465134171b6be7	bitman
73.	aee2b4616bce5bf8a75d029b262f192dd2d286827f99ea616df84b689e6a9608	bitman
74.	af40b4e873b6dd3ccc847fa714a5672d254026f4162baff8eb2bf048361b646c	bitman
75.	b012cc247648f644f14f9eb03fd78aff62bfea9ee008b49d872573ccaa71d652	bitman
76.	b0662422f3f68aa953bf82cedc1084c1cfe68546252c39932438435899a6cdfb	bitman
77.	b06bb44ad217bcdfa267c4e872ecb8e48247b53e049646c63ecbda83e3685851	bitman
78.	b0a2281613cef7d976092e7005b7b40e67935f6506424058bc85388242cced40	bitman
79.	b0ba81d72a723fc4a39a343fc67a4c02fbfb8caeeee5292dea87999a2b9f3091	bitman
80.	b102a135b53ddd45e40f8e9662536e89ad0e45fad27d20e6485cd796edf20540	bitman
81.	b1638b2601e170f7d56d86fabce18f7851449542491faffb59dd278b2468074c	bitman
82.	b1697843c021284f42a2ed56e56e2bb572ef33509c46e89bc7c9ef24d7a5113d	bitman
83.	b1775e8fc72c785322395a349ca1db16fee455e1e67ade91824fe67a926e2b61	bitman
84.	b1838fbc91fe6528f490c9d7db7a889777d62a759be53d23b84e9a6ba4bd0537	bitman
85.	b21c49778cece23d0096821050395a54ea388f32c1fd3478fba21c84ac8aa6ff	bitman
86.	b26be345df3edb94726f95d53398dd51c014420b9a2338b2529cd2473a7bc8c8	bitman
87.	b2d9c10b12c77b55ae9c875fa0c4450950e8897751afcce988074d324a7f33e9	bitman
88.	b30c1187fad2da93a0e6b1e3605ce0cce55ea7ff893bf9e751fb22db7d9198eb	bitman
89.	b30d37d08062487be7f22324685fcc7ca0c185d32fc110e456678ba1cd707374	bitman
90.	b35d539ed3ee08e9c095ca1588f5901abda19065d295d39a0e66fe984c96a43b	bitman
91.	b3b8f320ea6f30f8d4cb38959b2c240bfa71ee1dd9adce0added368a4343d29c	bitman
92.	b404e012584bbc1ce554cdac314db499c16af3d92a3eb4c9466130724923c903	bitman
93.	b41a917941665e5c28d3b82c15f21b109f1525d606fa2a915bc261ea4f7bc17e	bitman
94.	b44aa6acdbb83f90e7da27db15d3edaf1d377756b03705a42e4457db7dfef42a	bitman
95.	b4b0c622c84e7672f5421edf4ffbc778be7a147b2269a49c38a2e7c133e4cef7	bitman
96.	b4ce79ae9d2a9f0f748ec899919c608051eab8d4497ae5d88defdeacb8a17bc4	bitman
97.	b583cc06749caef229dbba5d1cc3862fd21ab35553512880fb9fb3031602d019	bitman
98.	b5cab3b87dabfeeb2f02181358ea27a007e2f67fb724b0772e98ca31f5ead5e2	bitman
99.	b5d02a6905d864246424769343a540005010894dc962fb9f10ee5f3c5abbfaae	bitman
100.	b6d147aadd8ec56a2d774cde09d7f310aba02d49a0ee8d495fe0659fc898ad25	bitman
101.	b753ae9514b2b142b965f570c604963bc63650a0c6e3fc2cf9e3ccaa1e1be3a5	bitman
102.	b7a7f9879cb689cf29d522b84cf2dc84ca17e6f99c8235edd4643a039ac4e4ca	bitman
103.	b7eff76dc88f530be6825e8aad3562d9ac4ad62930a2cc76b43098fc4bb0c718	bitman
104.	b7f4ae57d4bb1630e81ea1cd694fe9c1040cef1d04972233b82a3b0405dd008f	bitman
105.	b7fce170153fd95b3f7067808938b3397092b064917372ff0140a66ef9aa46b0	bitman
106.	b80bb23fcf168778a0032fec5fd3d76503cce54b2dfec38bccf8a9d49ab991ec	bitman
107.	b82182db409869a3b13d2cdc2c9327e96366d3686232060d3ed50cf215804396	bitman
108.	b86162c90f5ce7eebcd83d2ddc24c85823f08b466f22605311bf27461d82e4e3	bitman
109.	b875e47340c728f3ca383936b257b181c4d395c5709289b0cf188976449a1e47	bitman
110.	b89fb476a601c353f02298b8ed2e79e17ac583fbd062f6167fae07844dc19546	bitman
111.	b8d4273c342a8961cb5869c7e5daee547366fe63e13a63f71d4a4195029c711e	bitman
112.	b8d8cd056df71d4be6c3ed7feb31b8c9e5d0cce07248feb65f75f72cd0b7531e	bitman

113.	b8edc2af4cae67e1750269fa1d08fb693a9757fbcbe0618bf157c984b96e2129	bitman
114.	b8f7bf2497c921ffc6f7da02a48ad6ca5df70f3f941f1127b8ddc7f2f63fc736	bitman
115.	b94a75eb811de595b296cd237e6fabea2088e86b65db72a79483270b92ec26c7	bitman
116.	b9bcc5be53a9277521be2b98841741ddbdec43d6602fa99190a004c45b5ffdca	bitman
117.	b9d95d839d87a8a2708dc58401a1d1a0705d3a170a2d48564f0ce293d9a6c8b9	bitman
118.	b9e9dc47d5ec8b7efeed8d9202ddd7c3876e594b987108fb6ff4ffb375f0e25c	bitman
119.	ba0c8a8d1633b836d75349837698d04ba1294c2baa30128ca0f4873b4f255d2b	bitman
120.	ba5c558be0d98958ddf5de4e9bd6f78246ec5b795cb393ebb4350255f70c2936	bitman
121.	ba784059fa75fa4669b0bdf1f9c37846b72dbc475fd616e3d919da320585bb26	bitman
122.	bac6e395904dd6ebf40bd11457ce9343dac0c311650cba0da3372ee9c46dc623	bitman
123.	bb471564d65451b9e259d0bf825cc75b06b674a9c395e5fc8dfc2d62be223030	bitman
124.	bbcbf234fd09c842549c18f0b65ae73693ca947b4f5c507bef63418156b59c0c	bitman
125.	bbe4245ba0506708e8638e5b61a5f0ebce379be4c170a9a25d91fa30435d4cb3	bitman
126.	bc563305b280c480cdfe419495e4a887d3b6dfa0e0d20bd355552c437145abf7	bitman
127.	bc826245d47f907f9ae0e3d05511bfdab072c8f19817c03287107ba3160407c1	bitman
128.	bd34fe8ca1b5ebb1a71003ffde4d1dd082995e3a95893f5adee8c61c4b87a201	bitman
129.	bd6d65412820913859432ec2057050f36a046d2684b9c2054d7c4bfa51690588	bitman
130.	bda1141c6a33cd75b38acb228e4fd02c613c62eb2fe8092830e82f8344780ce0	bitman
131.	bdb9f41fe4b52566afd9a529259795ace86319464e453441549a23535a3a0515	bitman
132.	bdcf6b5f6e960c3b1a0f3b1fedaf8e8ae6f1907916da9003c2dbe1d5a4e49776	bitman
133.	bdd45100d2e8aadcb27fdb405bbb836547718827de6e59bfca50493a6e12a9fc	bitman
134.	bdf819317a9e7fe8d5c1b337625230391a91f30449979b9b786f43f5c840d485	bitman
135.	bf1140d65e51c9f43aa74d6f98f906097f29990df036cc5c4dd2b4620ed66973	bitman
136.	bf408af72187b3c2421f22fa114c16e60f96ea03fa9cbbd9cbf1abae2104eee0	bitman
137.	bf65cd623e43222cd1cfefe22a9c39c249e85799035f5f2d34fb89b01ab61516	bitman
138.	c0583726c03b8b1f8d50048299e40e2317e64516738c8aaff23196beb80e1726	bitman
139.	c063c96cbe38c8858775600588a0ced53abec4727ef9247496cc315d88800de8	bitman
140.	c0644acb2ce3f806759398dcc3f9241413235a2efa78011197a58bb7568cd0f2	bitman
141.	c06e1419142ad8cc50df3e724d24bd666200628e8f874aa69e51192e1786a636	bitman
142.	c0b2dd70848d5f5445778213d000e3e494ba3bb51b921089be108cba2cd4e644	bitman
143.	c12b0ecb2dd2d1b5b8e55b894cebb6be59a2e0d2cec744b1e765cac6b20f5431	bitman
144.	c134a0ac2809efa669ab3e69597873916f629aca0581664159c4d101a6adc609	bitman
145.	c1d342da624717a0f86f33206bcd634e13854c1a554eb3cdaceda4277b7b276f	bitman
146.	c22a74f1f2403b1f170d46074fced15bb84949445d7a1bc4911d64d42e07fc2f	bitman
147.	c27c90dd9d8ab05987ae319c6f18ee05acafa544d6ba4dbc62ac42bca7e4721c	bitman
148.	c286e7c24871a2b516ac2cda1bcb36ebac32d64c288244245afc6ddb401e7785	bitman
149.	c29f324219fc8a0f3f9fd8cfdad3d6615b028e259cfddf703725dee86cdf13e9	bitman
150.	c2a53fee5a1eecf1e57f0bea85cea0d9f3706ce6c555fb1ecb06bcb10d76a449	bitman
151.	0afac8c4e562528aae9b6f9437becc8f1d60c9e0e4a10423e32d9be9d53916a1	cerber
152.	0c212918fb29d3388e3f59e7908dbc9d9eb29756423a057c7b7da6792c6be73d	cerber
153.	0d47058a0bb7619130349b3b55ca5d63f27f52921a166e2cb8e7d3cb16828534	cerber
154.	0d71d94430949b7269352088da0d461182c5a758c3748c3941a9936109d3cfb9	cerber
155.	0dc32047132e00a0e7d8a3ca4f9c6dc05e6791225ba59752c40ddae896ac7799	cerber
156.	0e4cedceeca07589beff34be6d77ad69c5b37007a7a76fab6478c80a9a1c06f2	cerber
157.	0f62c1a40779e91a222fc8e84c7057a04b0b36a52efc2d9bae5c67821e2adc89	cerber
158.	a0cfd2ab95e6ead5f0bfe92e10345f415aaf5f3ca168d7ecaf380c25d89b215d	cerber
159.	a12cf05ce7737db8e6167828a7b88d72558ed5ea0c7b6cfd25296e051b759370	cerber
160.	a1e28ffa4418c83fd715e8d779c8c91e0eb5328e1a9c2e03162d1eabf78c4ed3	cerber
161.	a243cddebc64af8de27d7f69e30255b9084b73458e4ed3383ae5da9b36f92770	cerber
162.	a2d3b500bda8c0755d5aa167b19143f63740ce747c808518dec18a3df677e857	cerber
163.	a4366a9a3b9f36975c82cea26b610c7b396fdef2d2577047eed3db21b68a6e15	cerber
164.	a5ac27e721294aa91012dba81fdd42492b523ddbc9af0f28cb93d75b6d60d8d3	cerber
165.	a5d2a35b6ba6cf6f2556437bcec6c1759bf8e58377871aee149dac638df45aeb	cerber
166.	a6be1e543de9881760827d83295b2a21da38cf37386b23ee118fc3c5587bc58c	cerber
167.	a73cf1e44028b28d1f4c39053116314912051ac8d8874a76900ec85c98df3cda	cerber
168.	a7737475680681f861943c73a953d1b5bec4c0ee7056cb36b85f15606e65d2a3	cerber
169.	a77fd38ffab0ade6f85f547fdbf4d64d45a5cec87c75cc621397648d88262bd7	cerber
170.	a8ed3e10144fce5e64206ecfe66539bf3cb02d908f7be24f5cd65865be564b7b	cerber
171.	a941bb66032328120fe5736cf6ae501bac4b9233346188da9361616850b53b2f	cerber
172.	aa3e35aff95b7358d8314263dfeace5d8c92705d2c548d8b68a9d55b9864845a	cerber
173.	aa84517b338179c5464a50f8dcf809c4ced45f4e02d35d9726c81db0de98831d	cerber
174.	aaecea0f63c3ce4a5db461ebaad5d7ec416352e0101586bf90bea33a5d8eec91	cerber
175.	ab6780a4dd0488341ed72f6cd48a5153a6c4a7adb29df922eecba3017561bbf1	cerber
176.	ac1bec73ad5b11f4033456e78f4964b97b0c502b8ac9c0afa4f304a91dc11783	cerber
177.	ac5e6fef832dd45de0bcd265b5e367efc2687885b832feb7597127898f70364c	cerber
178.	ac6c1f7d0bd475d1e75fe0336f71d9e7da78347d9f903a9d4b0051cba662a349	cerber
179.	adb1746bba6a80a6c21d50e43ff594b01999a0c4026e1d556df7fbe41fc79ea1	cerber
180.	adcfcb0c255817f7a34312087273ab700bf1487b10dd488370590e7d22c1c9ab	cerber
181.	ae3b68fcf2e54309866a62d8ec1945128b2cbd1c870f4f6338d5da47c8f0f756	cerber
182.	aff0ef34bb73ea18ba5465028efb7c0c71a5bf89fdbbc28834e9761fbe062ffb	cerber
183.	b0670f2ef868cde0860ebdf2af96c1d343c9b8e0d18498e96f90446efce6fd02	cerber
184.	b07fdaef276d1ff247e0d7aa5eb4652fb845a7b08e1d717ee4cb9e7cc2df9765	cerber
185.	b0fff0adec9112c7042e7f9d103d5273ed4fbf09c03352fd56db7fc061875b01	cerber
186.	b1851ea50e4f46ae7f3d291124f7e6240749784263a8c944b3dab96fd68b0de3	cerber

187.	b236498bd1bb22ba7131c1ddda538f3ecd0169e8117b5a1e3e394bea5eabe37f	cerber
188.	b2f43d4dc2fb562b4db647b56bdef2d70141426fd50cd230a5a74b97b0d02f7d	cerber
189.	b368ec59bdc99e41201e24c4e3519fa28f81843b2a78907e55efbec40ac3fa1c	cerber
190.	b36d8b398b479a40eebfb3933de94b3e7fdf2790b16b9125e359e22ceddf9eb2	cerber
191.	b483a22bb87abb1a2bf75899b73481e0d6869afbcefdeef99c403a0abc0c6056	cerber
192.	b4f2906aadf8b14c5bbb24365ac696f7847e2185313cc840c4e78183067c5ce8	cerber
193.	b6cc9579c46ffa02d1ad2a3c772d59387862b8afd7870465d7c51c0ed5c2c5f8	cerber
194.	b7efc79811338974c22b4668ac6b90ed3d1e9415e1a3ac1895d7e4fae598abc0	cerber
195.	b9e2264fdf0c802c134bae5444d3721b0ab4019251772e7d574fbd3af3b2fd32	cerber
196.	b9e5191b37e437d954e6c0813f727c11748084453184c98b7da28eb3c0aa9feb	cerber
197.	b9eb9c5da8a69800c170baab49db75d359800b7c9f36142c7106762751871589	cerber
198.	ba381e07cb4601df717a821d4f57dfa22840beaa2e12e1c02eae71f649a7e992	cerber
199.	bab08fb7c2997d4a96e116ce0ca2de2c68b66e85de9343cd12dab9694f85b8f4	cerber
200.	bac96ae7b27c5c1cb072f195b2f57e6ffdfea0d1fe0adcd7c04e38ae17c1045b	cerber
201.	bb3fea80afe63d9861f415e53fd086fe7627306d15fa39cd58c8822956e0edc3	cerber
202.	bbb1ad47191bbb9c03b2438477d044dc6ea0014868fff5a08073ac1118894932	cerber
203.	bcef6030ce5a0590f1bb5a47c561d3df6220c852bc8bf67cb5e648d9a8eeb17f	cerber
204.	bd00a3de7e52016b34ed49f63583a3726d1122f6416b9e75df457157e33e6e7a	cerber
205.	bdc989744890072ad18234a5a63d1e3e1694056fe12896098527d53b120b0ef7	cerber
206.	befc59e574b7840388482f6a96ee95aa4aaf9b1edbd301686d49926482dc992e	cerber
207.	bfdd481f113aebbbb2f114b67411bbf59ade81d6e714477cd164ac737af806e4	cerber
208.	c2be437e13edce7a9d2b1b8fb96e2bbc9658013d2e6fc2cd3a66a3e697e5b489	cerber
209.	0f6b28ee75b58be8c8d4cb73f6752c42a063ee28a9dae1420c4c97ff65647a7b	cryakl
210.	0a9806946cb29ff0e95cc97e4693e5d527b987d653b965440541f29bf033e2e7	crypmod
211.	ab8098ed6fbdd6a5a33ee58279229d9cb76d67141afce64fb6dff63f73bfd88d	crypmod
212.	ad7cbf50e45ebc58aae9897851073bd3fa058e5685a63f78d300bac391929cc2	crypmod
213.	b9f3838c1cca2c5913e6c9ebc05f4f9fb65d6429c3bb2fba75f00dcc86ff6ef3	crypmod
214.	ae977eae68b7afe3acb760d853bc92ddeb9e75fe3bc926087f7c8f88ae6a0d36	cryptowall
215.	0e9bedc57f97bb2c7119ad4713b03fc9b10df09202fb7a237b610aec4687b736	cryptxxx
216.	b2bcfc4c5d1d60f7ea4298d32dcfff303f4db4b1ba89a8b6d24b7ccfe883e45a	cryptxxx
217.	a0c0f18d1c867c486dc5f9f68efb5488c16a63e7322db27afd9cbb496fab08c3	crysis
218.	ab9e0812b05d0f9a32affdf352a8d85e85b110d27ed8fe685931e706a1aa88c1	crysis
219.	b67083fbdf8880345b5e7eabb9bc257607958b2c61c2f6b70f158c5d10603653	crysis
220.	ab29528a67b660eaf1201ef51f5e5bbb818c2cfffb042bdef1270e43bcf1994a	dalexis
221.	b212ccea8a8579b9ed90782990b16ffa56299bec43f761e677cb649387aa4032	dalexis
222.	b8ce751d638af73b4feac31623b7f373ee6eaed6a8a0002215cbc69a020a8eab	dalexis
223.	c22dafebb84c22df623546a3e72886612a697a499ab10260af7267ecd37ade68	deshacop
224.	c20c6e9fda42e4bf453c839c874c6e1ca38f5dd2af41b3c4286c09e5fc34ad12	enestaller
225.	ab4db017e1e09c4f284a2f10a052a9282bb81043de9fa523d814e6e78f6515c6	enestedel
226.	a9cd68be689ab5f695e2f710f2e13fc54b779b15cabd5a7df306ddd87e6a2d28	gamarue
227.	baddad2764055fac7d717ca688777991f173c5cce7c23413aa71e63a16f85c57	gamarue
228.	b828059c7af40bf42c036a16e4c8c3f11eea6607beaec93f42a01ab53a8c5f33	gandcrab
229.	b55d23b9df8ffe5678234a2ebc473afb3024015c2a79dfef33a1824d08396139	jaff
230.	a84fc1196fbcbfccd726faab9eda5f562269708194bf1c3e54c15dcec82adec2	lethic
231.	b149ac6820820cc1fc8c722519b554daaf2c1cec4274d2224fbfd938dfed7ffc	lethic
232.	b35671a0da9261e972e668d2b13cfe654d935eed7ec9a258e1ea80069cea2fe4	lethic
233.	b52dfd9851c8d9e73cf644e019f09c73b4b8b965703933403d68e7c38cfe9bdc	lethic
234.	00ed72ffa97f75e1f4d1769e25f94614dbf3fe2a468361b3511a7c0b10ba9038	locky
235.	0c96311fbd6dcd5af7dfa4875fc72beda0c9e1c7470ec8d0cb54041201f94708	locky
236.	0d80447ad564ffd0c40cc71213787f823de1b058dc1aa856aa67f438dd51d537	locky
237.	0d8ab1be84c9ce7d3b167b344cbe5d7572448f7f07f01c7c4ef7ed5c8b68ec98	locky
238.	0ddc0f51f16a49c6ea129b63eecbd2001ddcaac050f595fca5eede491f7a7693	locky
239.	0e18826a91eab4f024c91d5c2b3da7b825747d2cb53829e13da98a3848d883fd	locky
240.	0f75c08edc81483acae170972d3f24dea05149295773badc126a61961525c251	locky
241.	0f9ca5c555ddf4b5b29573ea1a513a69555afcfd0b1d3fa8f441bc6991bce543	locky
242.	0fd0d74780af9c9c15ea53bb7dfe8bd1f7bd097bdfcd5140a4a9f9e78b7ff79b	locky
243.	a1596136fe5f12521e5ee1bc05429459453ff8cc8a12bf35f23738ef1c0ce8eb	locky
244.	a2b37fd7fbdb708041f257c1004ad0200937817bbec3f54f3ee67865163c6658	locky
245.	a2f39c1650e4f237741dddcb49e1ddcdd1d5d7b7ac1dfc007049e5051bf2a3c9	locky
246.	a512406a4a94325a618cb0a3d68393bca6c714024c612f0074529e4bb5ad8071	locky
247.	a6669aee7aa34ca73e776e8df5083c2934cabb11463a72702658829689252cf6	locky
248.	a8b7cda73fc3e8078aeca585af9dd748b11f70603e91957b1ef1fec260e5842c	locky
249.	a9c72c30ffb007e04926f77ba204fc73c7f72693e9afb4cd152b4986497a5000	locky
250.	acaf8469aad50639040886d6f5fc9681930351a233f1e3db0f9f833033ecd8f4	locky
251.	ad2daaeefe92766ad11745aa3910b127bde4428ae631748b50c3a9066fe135ff	locky
252.	ad40440bb8b4e443c9537675af5b7f588c4289b9cd37f21a91cf2d2110cf8f74	locky
253.	ad4940d5b842b9c8008fd5ce35145c8e4a6465d5e80774ca203d136496ff51c0	locky
254.	adcc6eded5dfdd7a5d79dd7a9c2294b41aec2f173c09d3096a7516d49c02c86f	locky
255.	adf749ad3ddb0bd7a67e4ca0ec549a44b6fd22d525a839101085b1111d669c28	locky
256.	af3568e48341b40d5006a12a11971344577110afdc42a0d8b53f62453b0b389b	locky
257.	af7797bf0df65314f3173e06b114b0498ee0d76c35a243376d1bc1efc4a01347	locky
258.	b01a05fb7e4d26ed9f760fd08658024a4265302997b6e8456a606d042e6fd13d	locky
259.	b1282ad8fcc12f9457fd9344e6dea01dd14edac7c0627482255bc9878c17d6b6	locky
260.	b2444acebcb63697fc73c6ff0b462a176e277d6ed7fa127f28c203853bcd4d03	locky

261.	b428e5d84776ac342681ab069cdcf0585b62868a6407345b508f2c459f870a71	locky
262.	b4b7ed56b9243880006e6d0ec2429831332d0ad4d71baae2add373768945c630	locky
263.	b4ffc48d7ce966631d1a7eba54f91047c979f53211ce077f528068adb2140cf2	locky
264.	b5064979527714bbbd0558f1ede2a47072374c36e0e946ce1855eaafa2112b7c	locky
265.	b609d78e860248f3631a7093b64ee0abbe90768be4de122bbf579d127426b49d	locky
266.	b694bd9859ad8707b0326ffec045c30699514d0b0eafe374bd125baabfc1c38c	locky
267.	b88f7b654cf46aa20a610cb0aa280ca80f6d5128fd079cf719a151969943fc04	locky
268.	b99b4c7847b3a31454d4a2a2caa29e40924e2279b6cb65c900d261c37525bdd0	locky
269.	bb2073c2575f6e253b63a1ffc99ada1ee60278504eae0c8e73ce273cb93cb516	locky
270.	bcf6d56f585b61ad22094e7c893d12d733ed7623d56c53de97bffe43041c5814	locky
271.	bd6f0b5056e3a831337d1092342a207bd15f76706d5eaf1bfe6c80991eec196f	locky
272.	be6143f0d5db62a2ce51e5c85b207632979e50098163588b2f8ba3acf6e7b67a	locky
273.	bfe1bead58ca164501b619a93a93a91dcd25f147615e181ebfd68f7eac64ba13	locky
274.	c11b9d1ba0badcc063eb6e60894b7f4f0932e4f73d037f05e06c80d72833b328	locky
275.	c1245b403a345b76a5f4aa82ec336842b2355977055c80cfd2417d8184af6eed	locky
276.	c127b95c9d4710e862ca8b477928b78b43bcfb6655c27462a7c644daac1e3c30	locky
277.	c2e56510866a6e038ac723a3e5a2ac66b14f407b91886077727f622f561164e3	locky
278.	c35f705df9e475305c0984b05991d444450809c35dd1d96106bb8e7128b9082f	locky
279.	a1ce492b077c0d0cf6fa362b57e7f6b3b9d1ab7ad60541a12e9eeaa7708b9af1	midie
280.	aa8768bd3e213219bf17a0e0d2411f5a2f6ad1e60ca55c489b72e27166c49f01	mikey
281.	b6144a20dc6b4656551494d6fee1450abcbf6c4e30badf5765dd64a254495e27	neoreklami
282.	b86e5f28c965802604464067a73560b16a710f6df599e156a5aa23e2054b6295	petya
283.	b8fe3a953887aa22b4b3d1d9870474cd54ace319ad678fe1cfbe860bb0e5ac9a	petya
284.	0b70b21bb62f9b9dbcde8da791b84623afec45ae566b4a0acfa53fc4b861324d	razy
285.	a4204086b787f92fa432c800df7751e5a2f64d5b7c5df6be39c21779ee5b2747	razy
286.	ad1a191331e87832455c4ac315af8d01788c5d43c00ea26f80c54f6145d790a5	razy
287.	af29893ffe76fb286d51655327dadae5f9f20e5dcf4aaa6a5e4f555b91c5ff73	razy
288.	b124a1b33d3b2349c389ce3ab77603dd39b03023290bad9ae9ce6e5975767727	razy
289.	bb37f3f511683bb363e6fe62a2e83fc74b447646e1697d9ac2cec3e7cfacf984	razy
290.	b3040fe60ac44083ef54e0c5414135dcec3d8282f7e1662e03d24cc18e258a9c	saturn
291.	b70cd75e503a74f3197429f1562c23c92bdbbe8d803ff5963ddfc25f29652174	scar
292.	b7c9e8a46426aa8a56e8a9f6d7e8b7db497f533f38c281e45dffe0737f1e557d	scar
293.	bb96b88b03cb4f54c3749e6ac9eb54785438e04023f38a93ee7b2f908b7446ab	scar
294.	a01e5f9bd86868992024f55146e1412b593b610d330d6a200b2748d3aa3cb457	scatter
295.	aefe62e7705678d90d5b11927a8cbd4a04fa32971b4df6e36cb8199b59bf13e2	scatter
296.	aac56d25685a1b8536dd5efeff9fbd8845da20693affb33acd67724ae998a6c3	shade
297.	bd4d300f37b230e7c7578bf453d96a15553b88868fe13d4ec2d94bfce4eea567	shade
298.	0b2cd15983e7475d8a27023cb5687a1343d4d963e2ffcaa6538d9337ed4efa1d	shiz
299.	0c2519955f1bb8c552b66e6bcab181f6d593db2d5f1e679d5fd55e0dfdfeb762	shiz
300.	0f770775248a08598e59110eb9ab2630ad935bc0df2c2c643d3211c3abf9123d	shiz
301.	0f7d43fe5b8d69752f0c83ef5aca116dd7f0beff0273634518c84bc7d4a37a3c	shiz
302.	a0870dfdaeb889d4285f4bba81fe5bd4fc76bcd0ceb3925d39c65298da5e4bda	shiz
303.	a12fc5685d9a09ea492a75a0764c61b9d84aeb4b923bde62b9eb96608ab205fb	shiz
304.	a2b189b19c37fa23798ff1689c076f0743d8e63587d397041409cace20c87829	shiz
305.	a69d02fe52333fee24b5d1f8aef92da8877c8cfb9f32f7b7117b7ffa9db37c8b	shiz
306.	a8a5577bbaac9aace0bbfc6d7f57926a171cb2d5d112d6066ac9ea9e1f9a6811	shiz
307.	ad71b68f7b8e2a7ea94354cd0733c7fc1aa29aff9364175f48823f5dbc2b72e6	shiz
308.	b2c335fd2b43230fd8bef4858365fb18daf1054c2fbbc648ac97ed2a6f1bb27b	shiz
309.	b3fdeeacfae9268f213940605f119e8e196287e8ab81850636cee9111aa2cc61	shiz
310.	b4d6af0d624dd46851de333604316dbf55f9c36e6cabc11bdb2ff5c201e19ba4	shiz
311.	b50a5eca4c159d3e2a1a81b73384407238710bf5556c86282b4e90c20b22667d	shiz
312.	bd3c8f5228ec773432faf4be5c8481673479a603882bea8b3e816df1367a5482	shiz
313.	c1df72723a7b8ffe360f8d6290fbed493674e54783f885acc2d3129a704d838d	shiz
314.	c33c22542dbc59aa66b3346b8921086d95ad4ce0daf6fe3905605ce02337f43a	shiz
315.	0fa6afcf6e176443f40a957d536390b16863490e7ebbd67d56ad9a64ab089fe5	spora
316.	bba5c1b169c80cd519c00b35fa4a0bbf209bc7f763a10c240613df6f44349339	spora
317.	0ade3100d2afd6d9623834b2c99d36f2b14c0fa154f43f72989b5b6ba6fe3326	tescrypt
318.	a1db671d73239ec6d62eaf48c6128daf78c2d2ee5f9d54c4752f6c1b64eafc42	tescrypt
319.	a4156e684f4335d4663e6d685c78e2a6fb8374a6e1f38f3ed970a9c5ce1b0211	tescrypt
320.	b55bfa22d913d54bfcf39a749ce829a7804c709b12821fd7fa1df3dd65975f0a	tescrypt
321.	c10ddff904d11d0f4ae719afa2ba37fae00bca01deaad5c8237536441f39af90	tescrypt
322.	00d340cfd9c80206b0d282187ec15d8f93164cc1f8e65809038ac4d74b3ba0df	teslacrypt
323.	0a0adf21ee7edeadec0d6d93944e4c01504d53cfdbf466fea3479cf5878d812f	teslacrypt
324.	0a47655d725a5a91dc401b51a67e69ba8c85cbe7ceb2200d7496d5bf5aa58f97	teslacrypt
325.	0a5b115a930ea47f8d37e4c6a936dd4713bc3df0b401091c404b0d95caa4a7ae	teslacrypt
326.	0a605935da8dd95b37190e8a083b8357ef035e4850d335609ace78d2cc93ffe4	teslacrypt
327.	0a630522fccc4341ce0ea4b21dc490a2f32bb8f89504758480e03cec55962863	teslacrypt
328.	0a63be6f6c13d275a14b734111b8c13baaf06e6108ce8c4cc0a4407f6ff5d18f	teslacrypt
329.	0a6c9afaea8b59b99364a50ef5391f54b3deeab9dc8c8aac40c9b2b3c5f681fe	teslacrypt
330.	0aeca2acc2be0622753e2762f00955d74123530a0f0fc2e2f721a3dcbeab5542	teslacrypt
331.	0b0827ad2ce8b61329d1ff84ec763b151e49c83312a78c4ef46909a33068d765	teslacrypt
332.	0b17c081060be4975e1fc41f929b27c30077b471da401aef273ebb96631f1e96	teslacrypt
333.	0b2227232b786974f45d9cef9fd49ff20f581e301f2ac6406ac620de30f1285c	teslacrypt
334.	0b531bc4f0f4ea0913f6ec2fa01873e2efdca85c86c72d5e8614f18b5107c027	teslacrypt

335.	0b5b3ffaca1402eaf6efc8bb7b1f89b0745c6e32d9eee0dd83a2d7025401cf89	teslacrypt
336.	0b907a6a20511ce40020ce2601d25546ee844c29ca84416a6e18260159e561b9	teslacrypt
337.	0b94f23319dc90c567f55b3a6751d3df5c31c3a715fd9df9f9c886db77d89043	teslacrypt
338.	0b9715dba8554cf431eba283cc182a5f286b5b8a10a83bbafbbf0b266bde836a	teslacrypt
339.	0bba827860c9c3624b69c2611675170c78e3a64f44a775fe8e0a3775ff994022	teslacrypt
340.	0bbf03be24dc308e84ed73035601d5d145f280e5c340c21eee57354b46092d7d	teslacrypt
341.	0bd92e51f473e1320353640fa26a9a543ee71a68b689aa54ec9df5b18e005188	teslacrypt
342.	0be94e1cfe8deb7bb41301aae644cb8a2b5898f4bc50f13eff1b3ca45e8a47d0	teslacrypt
343.	0bf3cfe6be4ab21533046a289b2f4aedd5d1e4c48ed9a2a0d008821db4988cd8	teslacrypt
344.	0c00b8186de77029f7717c745e5e2afe39007598c72dafe06b35d0827ba2323c	teslacrypt
345.	0c06ef68a001fb8d9d255f6c352086b60fc5f26a44b1acf72717c3b2020e56f5	teslacrypt
346.	0c25208d0eb1a45fa8692677ba7322cc42109afd8bdbb9fc4b5d9f24adba99f8	teslacrypt
347.	0c5991cd96c09101b6f9e0830237a82f892a6ebf80312695db08d6bfdf187fb9	teslacrypt
348.	0c64eba4607b9fc93957fc1d55dd330b9d4a4608eb3703209d8914d06159ca13	teslacrypt
349.	0c6adfe2e692dcbb051d58643ba1e6b5f3d5488386708ba97acd4911e525a51e	teslacrypt
350.	0c747830dee54a4d9a50b96e44a3ee9172457e801d954f77ba65a6d6461ffb3b	teslacrypt
351.	0c9e122f5674c5b6e556b5df501557248dfef05c1de0d7c6abd5866240e2fa12	teslacrypt
352.	0cb03a83b88b62025f2f77b1788e69c10e685c54002486b487b0b24ae98ec631	teslacrypt
353.	0cb6692d0f8a8bd856f58f1c24cc1aff43607d61e9650fa5593b2e04fdb7a6fe	teslacrypt
354.	0ccbdb60f44a3af9b6b8781092d04067ee19bf64971a74a20e17c44536d0fc9b	teslacrypt
355.	0cd3f52c0471a21692f3bafde0d292a942a80e1954ae169a4df7b074cb92c2be	teslacrypt
356.	0cdd42ed61452e28c50bba66619dff45ce170238b4e22542c267bca514edf2ee	teslacrypt
357.	0d2a045bba1da9bc7ca16797aad362962a19efbfb75f809054bdab13d9f3c356	teslacrypt
358.	0d30b68435df2954d0d2358bf5c955bbd2a0693401cb291742d7dd57df434f14	teslacrypt
359.	0e14aa56129ed4ab1929037794604d88dc472a6378492389b44a618e74a87e9a	teslacrypt
360.	0e4e2fb2c2f9db122ea453d2e70ad6de5d3cf3d22f67009b8f70631963f5d3a9	teslacrypt
361.	0eab309a8088b5fe8982c97f43ee717e02fef4968f5787a1f7108d793aaa1d94	teslacrypt
362.	0eb05e3a678920b8d1770a036a98bf283931a5a02e9d3ae893149a36a533ae06	teslacrypt
363.	0ee41942ca5fac9f1e8525cbf5d53479ca8f2ee6a51e394ba8f047bdb919c944	teslacrypt
364.	0eefa8bca3e42d2edf5745c4274c51cf42457932467a614544274c8bca255d81	teslacrypt
365.	0f08244193e78cfb3b373694f3def175bff3e94a1d72716f435f58e3331a942d	teslacrypt
366.	0f2fc2bec5577aefc2bd2ae7fbe2439abf1538f1d508c862537da017d265cf98	teslacrypt
367.	0f395855da1758e4bad73afe400564384ebc01093578bd8270e16dd461ff4407	teslacrypt
368.	0f6e778618fd182e8f9e707ce88e6dd89c1fd7ab14f4bed56a425eb824d9e9fc	teslacrypt
369.	0f79ed97257e4b84c1e0de8ff0187544dd2fb22909df4fc776ccfd379dd007a6	teslacrypt
370.	0f9ae792f0344dd086241ecddfe63bdaf7224f2e89bdfbf2182775ae0c344142	teslacrypt
371.	0fe65bae092b7e50ee2a43b786bdf089df91bed8e9fcd8ee5ae44a3522aa4127	teslacrypt
372.	0fe970c1d7f98fcb5096c31cb6173e208a252526a3cea3ed4b520a47d32d8f0e	teslacrypt
373.	a056cd494b445b135bba72c42f2ba18a801e9e657d564c4f23c3972e87502317	teslacrypt
374.	a0658ddc1bc3d40abaa88e4611cfddb43c17cf4da70648c1c11f9ff5797c42c4	teslacrypt
375.	a07a769d70eb06d7650e681b375c1c8d75866452a7a5b3da1fb20311a68e7f81	teslacrypt
376.	a082b3171dfb990cdd22cfa3c0a084fd0b0226d207e35392a9575b5b3de0fde7	teslacrypt
377.	a091406768073fc305aaef7fe7c9e349469ec8584117bb405438a1cda4048039	teslacrypt
378.	a0ab8b0f923a42f755f443047007a8a9e7b71eefea5853efef2d02b3211d9d9e	teslacrypt
379.	a0b10c500c105768092721a1d428c9a1b80359a6e92e9420359a05f8ff6bcf73	teslacrypt
380.	a0d0b84db349123d08c2fab691df5bb5d6b3e0a44372e75beb9bcb0e663ecb96	teslacrypt
381.	a0fa20d58b554eb73dd036bcd9e12827024821e2c8cf1f38f24b64081c96c3f8	teslacrypt
382.	a12e2c7675613cc83546fc0dfdee1acab5168bbfdc35bccb874ab9c195feef4e	teslacrypt
383.	a15f64453da067a1ab584479276fe9b63f8c6771b98db0454799f96862873646	teslacrypt
384.	a1692c6ed1469a510a8d8007f3388b63a10bce006173a15e9c129b33f16e0e28	teslacrypt
385.	a1a1e42dec95b6bc1fdcd9a95a098cb9305d83634d24a4d589dcfbc30fe2cd60	teslacrypt
386.	a1b44e55276545d3e54e8323198cb54dfbcbdf20e0459729bdd34860ac68dfc2	teslacrypt
387.	a1e6f7205dc9b76d1fc28657accc758f848426b8ada4ce2df3e9272aaa38b09d	teslacrypt
388.	a1ea14c9e82d2169f4960d30b32334029f8f44cbb5e3a52b98c447ea4d5b81fc	teslacrypt
389.	a22d2683d79d83335cd43142fa1dbf53c5ad5d3eb17b02b61c175cd93b56b6cb	teslacrypt
390.	a28da4394c228bd37ea27d5922fd0131e4b6ea911657925c93c62b6ab0b352bd	teslacrypt
391.	a2981d4fd4facd7bccd02ffd4be44267156eed0e3f56f4365c22383dbc7515a3	teslacrypt
392.	a29d1bbec6a7890eb521c3d1e28274980545ee98cbe3895bd40f81259e6d11c2	teslacrypt
393.	a2dc35392fcddcb4a3b422d2b94062557330467d5deeef23c0fda4c62fbcbfb1	teslacrypt
394.	a311cb236d6f917e17dd0962e4b4b1d4f5a5186a19134f8fa708408e8411a3f8	teslacrypt
395.	a37db1598db38aaee321d0765e683612e5c49b1ab72bab74d47ead4e116dd84c	teslacrypt
396.	a3878e4b411e4eee88c977d3bba364d63cdf5a6efd210986b76c96f478a9c0c4	teslacrypt
397.	a3ad2144cd9867e3e8e89e46b4a04cb96d9afbe368c9016a8dc22d7e9ed09bad	teslacrypt
398.	a3c4672e2aaef1b443636317c4e2e3f90a428daa05a0b6a58fd594529c7f62dd	teslacrypt
399.	a4030fd0a699e66d5a9176fd12ed86a8d762209240bc369ee77533782030dc78	teslacrypt
400.	a419000e72f3bac2401800269d2db35e7a232aa7b2ed244682d0cc652723bada	teslacrypt
401.	a469d0f8159485de5b7b2a6e3c558bfdb217addc5b4b248335faf304d3c49240	teslacrypt
402.	a482dd842e8a6651169a0fd4c9f950f83eccf56f374ee675e82b8fb11871456e	teslacrypt
403.	a4dc7c9ed4e2450997a0f23d1329b958449eb8dda41a086d4ab2c31efab0ce8b	teslacrypt
404.	a4e87804412ca6af5c1b8559d9b754a3157e720a055058af45fe5e9468c602e2	teslacrypt
405.	a501b4ce4dba55c2be873105375eef443aad963466a0976500f9b7e96ef48b80	teslacrypt
406.	a51b3358ea9ec63e214dfb719c1a1788a698fd46648f1ee43998feeccd282383	teslacrypt
407.	a592775e88939808c3dea0241933cef9a9280b2f58612dbea640e60d1985d79f	teslacrypt
408.	a5a2f64d60d24aac8e5b9021263fe0951788fe34204d07e2732038ec00954276	teslacrypt

409.	a5d04667c3927b12a8f9f8b6af3697eecd85aa92e8560a4ba72ee71174097a0e	teslacrypt
410.	a5fdf24f46cee09a6ffd0b10c7b6764c44fa971554446c39abb35a15a5624279	teslacrypt
411.	a620e21732f43a5f0e5e74d867091f8b06739ca29e4b8d997ae936a305766b2c	teslacrypt
412.	a6794a139d3fff8ec52c4f47410295b1bb64ccb91793a5ba76b34aa7c6005867	teslacrypt
413.	a68a62ad0fb9c2da29c0e25963b83926b462936beae080721e0187f67a86f988	teslacrypt
414.	a72ae60effad2b9c93a722bc4337360f0c431351b881ca9ff5b4c14cffde8169	teslacrypt
415.	a73d559624121e53fce71bb6db797604b7f4c063ffd2d64e6af94c418343ac74	teslacrypt
416.	a77d7a48b5fef28ec95f7f727d0976e345622e41166391808e4c884e710ed1f5	teslacrypt
417.	a79613f986a715d67eff4a551c1f0aef9a18eee7ca34fbb022b2ab4cc0eec89a	teslacrypt
418.	a79c402f39c2e0b20dfe155c546b13385e9b8e7475d3ec0390729127272d785d	teslacrypt
419.	a7a545b2e1ced8b3f5c6446258bf6bf0f2d5369d7ce1249ccbc4dbe30176b7e5	teslacrypt
420.	a819551a415c9bac02ab6ca5bc135685e33da88e80dde7fd100bd8116534d7e0	teslacrypt
421.	a832d222e2f308b27e9b044f4656ad351738786a0b8a0a9bb456fe32f6686c50	teslacrypt
422.	a839a01d93edd926f9c9977e1a1315a3f978a4478fe9837cfb8a5bdcc5495fd4	teslacrypt
423.	a8654b7e9566cdf05f2fcc2a3921b010941120f34376253af4f4b21a54766fec	teslacrypt
424.	a8c563c5944da190320bac2b4aa049a6ce87bede4a2fcfb3a1882de1ebbbaf87	teslacrypt
425.	a8f8dee553a5b9bb3813d752e1ac05fc59f2518afac76adaed823e89b20965d0	teslacrypt
426.	a8fd0915099c50f9ac8c0c6d70eaf8c5ed3ef507bdcc50992335d18ed4d976e8	teslacrypt
427.	a9361d2d8f82ab3e8737c2afc9c948389c84e1b673fba0ad894daa8d99a993be	teslacrypt
428.	a93854e69170b7322192e1ca994f57946260bfa4152d690bbc346fe332b4c186	teslacrypt
429.	a967d0d51b160ea6c4741af9e2e63c54827f88fb4f7db73eb623766e42e5440a	teslacrypt
430.	a96b845e021a47f4dea43a15884f448e0d4ced583373a3e70479bdc89bc4b64d	teslacrypt
431.	a97ca82c265cfe643ab5b66aa19c4c57860b47ecd96545028be296da5d57aee0	teslacrypt
432.	a985f0532f82ebeb7c41604da5f4f54ac60ff0306dda6042d746112e68f98be8	teslacrypt
433.	a9a97602e21264b3d4d621259fa0ece954c36a27793eb65337e391c027d2c78a	teslacrypt
434.	a9c658bd2a3f746bd54e5de6cbcaed1e2a687ef65ae8277cec85a9d13b421759	teslacrypt
435.	a9d6ce44b9bb0d65ab301438b48296dfbe686ba71490fd73d4a39300edd31d58	teslacrypt
436.	aa0577a1dbb79629f9c09642bf40718a6732232f0a7453e5cd85708e3ec0f34e	teslacrypt
437.	aa16615c6a85245b6eb160987ddbd4e16377b892070a916f5485f26b9c27c4ff	teslacrypt
438.	aa1c836db5f1c1bf99de1523d02229d9c25eb5a283f18f4978bfc57f4faddcac	teslacrypt
439.	aa36644d6e099cd3ff0f5d5f3fbd1590641415a69a9bf1dc58426220c6542a28	teslacrypt
440.	aa48f73330c61bd87dc50f8e85e337067cd0543db774f3e5a3bdbdb4d1c5f838	teslacrypt
441.	aa963a73589a70d04c031316962fcc5fa12ee27ffab81fc7488125197318acf4	teslacrypt
442.	aa964caab52edb4f277ba778098b672c7c84ee83a20714c6e4bc43a68470e558	teslacrypt
443.	ab18af9382795db7a9b8deed41fd348179452d4868b7047f92fe9eb1edd13ec2	teslacrypt
444.	ab3ca694e5c990974f69b162359ea4f8f0923e80ec3259df4fd63b4b4e22a3e6	teslacrypt
445.	ab5066b545c65a2d5b8afc05a9424740abd96babff85e63fa7a146b4f449456d	teslacrypt
446.	abc2f2ecc626bab2118b705e726473845a2ece5e4a42c588b23dd305f6148cb0	teslacrypt
447.	abcf2ceae5ae0c3d2ea2795b2906d28f5ad63940774ce14e12f55b876e4850df	teslacrypt
448.	abcf5eae6e668213b6b7a49b19aac7394df9753bac1890107a7cfdb65c2a73d7	teslacrypt
449.	abdd1436508d264cf44bb883ae4e8ed124973c63e09d0eb19146b3723633f863	teslacrypt
450.	abde6f2d2d467f19b3e2e1d8bc724d8fbd53d4c0f08e12bbb081b067a7249cde	teslacrypt
451.	abe111f9f6a52eee75fa7e3842a24334e11fbb3ff8b971efd011b9fe64ca3a99	teslacrypt
452.	abe529e0d798646db7fa427071e43b0009569394298a587334fa172a4c191b3f	teslacrypt
453.	ac574e6017f8d30e95b4716855cdeb38062c4c86eb99caa38c5681d10c3a4616	teslacrypt
454.	ac64a45acbec1208d5bebdef6f03cd2a29d049b5b12024e87de279352e1439c8	teslacrypt
455.	ac7549ea40d956213afd26e5cbe2cdbd3a09f625df05b268f2c263e765e1058a	teslacrypt
456.	ac945c2e1307211cd431ca8e1312be1e7dd842fcbb87e85bcc9d72294ef32704	teslacrypt
457.	ac989bf33ac7c91a409537be0a6d699ebc4e15a906d2d17aaf3107c41df1d376	teslacrypt
458.	acb7a485f764a5944e75e1d747b98014dadd7717bb02200bfcbf5bf6f2949440	teslacrypt
459.	acd02c0df2a3df67e7207fdbe91c83af6106245c2991283e589d0b0d4cd2f2d0	teslacrypt
460.	ad2e0db44113ce393188db553f348879e3a93435f178c2600e1dba7ad0459dc9	teslacrypt
461.	ad4bd8554b9f1087eb8ce921430bd8a32b90499ce186ffc2d16b8c598a52752b	teslacrypt
462.	ad4ff31b5565f4e4ad73813a66704dc985d90facfb5c6c769fda9c165b37d8b5	teslacrypt
463.	ad69b53f8a9284605a7f1b8c6d5cd9d5cfd427772b1fef0e1302ef8af1b4dc2c	teslacrypt
464.	ad6d1a070dadf8c009b08d6cd747ac1adc3684af8918055dcb84988e0a8d5e11	teslacrypt
465.	ada7b7ec70f724dd17dde617d6a80eb64d6b40835749ccd36dcf677cf6fa063e	teslacrypt
466.	adfec042abad141bd9bceea35aca1a0b1f56dff195601cfdb1bfe6c749257a6e	teslacrypt
467.	ae06f374f9d7c5150b4db6aff77240ea602c84916ecafdcc739765d525678dfc	teslacrypt
468.	ae295ee7804a316153434783d83c2fbc63954a9384e2343a349be88a84e4ffed	teslacrypt
469.	ae56cdd2f5bd9b2467a865c7dc442cbf87f3564fda95b7aa86e14c1655a3aa5e	teslacrypt
470.	ae6f76c6d49b7459bc5c7e9c92bbfc2114ed1a4bbf4fbc555f91e69aa34b92a3	teslacrypt
471.	ae83e45a8b28a09eb5996b1ab2b8dbd3c70cdf54cee00233d98957b764a76c53	teslacrypt
472.	aea5a802d00b9a8faa1d58eb28d90006c5f688f956f7e8a18065289d84010b37	teslacrypt
473.	aeb78fa3231ac0557a7784ca01e2641b7746526e081385d1e2d3fe84529f0c21	teslacrypt
474.	aed30d6c20db1ec5ab215356bb9077038aadea5cf28dded5a6a52c2fedc57c45	teslacrypt
475.	aee10ab1cc09a8c441d561933cf160de7b251bf3e3165ec1baf5edf0ca69ad5e	teslacrypt
476.	af20e170ae290df4f0b34909c51d18e0ea1ce6fea47adb29d858df42d4d50750	teslacrypt
477.	af22cc42350fe049f56569cc945d3cb5017c8de66052a0e6419957873f403468	teslacrypt
478.	af907d68de9eace6afb5a430ea672a6869854b4e15ede2b8d8e236ad86846eb8	teslacrypt
479.	af9bc0f170e3206d5ddce0575c50097140d77e2ccdfdbb738e7ff0a3d8265819	teslacrypt
480.	afa9e65b39fc6a101c902139888668970a7ae3169af35f9f2c8ba6bbc635dd74	teslacrypt
481.	b00870f757048ecb093f6fff9f2e487d932ff38ad41ff6d298082e6a0ebf2080	teslacrypt
482.	b00e6d1aede356c6e40eb4b48320af33f931b79f9dcd833c57a6becd9ea341c8	teslacrypt

483.	b026ebef43adebb612535b065cf0056ce4e9dcd13c0372cb4d85060348493f16	teslacrypt
484.	b04f17c1d93ca085b43623689be0bbf6eb6d9c725b47293b31054d4195e56c34	teslacrypt
485.	b06fbe4b98d36f20bb218d19b0f3bdffc26242d6e95c282122d70f07a0ed3f86	teslacrypt
486.	b09726e2efe4dd89dacd5ee3d541b6abc25249acc4c59aea734551309277af6a	teslacrypt
487.	b0a5e0a51dbfcb0ee727c879705f2d1a50cfc3e3e178042ab44372cda812b1dc	teslacrypt
488.	b0b06754649b82b3b5c5e2a2ac10a3d878bc2c335b7b4c1f430c6d51bc2afbb6	teslacrypt
489.	b0b523ce0cc39125f7209aa8b27ae1a83b7dc769fc09ddbf6b721b3f005f479e	teslacrypt
490.	b0d31165de9214b27eecb013a6c1693cbdd091734b619c8cc8be0b9bc5bbfc0f	teslacrypt
491.	b0d49cc49e5a7335dc43747fa26222561924f54a250c3a3bed5c1d1844a66d1e	teslacrypt
492.	b0f6596b28e0d1032c379593fa657c27e59b85fec3e72602b6f0312b535501d8	teslacrypt
493.	b113bdae157a6cc15762b83777b178f00b315e484a502d4f9a79ba7636862328	teslacrypt
494.	b116bd17f52e19c2447b5f7c5aea7f73788252d7911d347600121406bddc33fb	teslacrypt
495.	b128a3392950fae05b8576bb45306432322e7dc22d089903f20933badd1ec0b7	teslacrypt
496.	b153f05e4f95f00ab475e5fddef258bc01a43f0286666a00ff317fed510a99df	teslacrypt
497.	b169a47683e5ba46fcc12952755bfc795856981f64e4dcf40b2ecad84869088c	teslacrypt
498.	b190ac8251a506126209208d4643852d8b98e681e020b9c76e06ad76672733c9	teslacrypt
499.	b216ee7d6f3190a898e19f29d9d26ccc6848338ca277407f7b48b386c1363294	teslacrypt
500.	b224b52301f5c6225d5f180d1f97dcde286c723df706314350505ee03cb4bbbd	teslacrypt
501.	b258c7c81dc771bb435afcc1eeb03489ca92fe49fc4c80aa94c4254e56fb46d5	teslacrypt
502.	b27530bdc09a2c6dbc8a249aed714d666456eee4b5a7dc3d0a8f7a5db278dc3e	teslacrypt
503.	b29f08f624a6ffae2f9d00e510f6fa7622c63bcd0b394d2a5235b6d66712fa92	teslacrypt
504.	b2aa4ddc259a0d00fb7b38cc0792918e1b655f64b4d12291561ec65dadd24e95	teslacrypt
505.	b2ac6eb697177ba4569176426216b4b9d8a061cb95b584e98d8753d6209ee6a8	teslacrypt
506.	b2c034b3483aaca882f7c533acebf7df5b0e7041e4beca271edbd7537bdeca8d	teslacrypt
507.	b32a50dbff7e550a16db395477f622f2fea2b2f41675a23ca0afd54e169db706	teslacrypt
508.	b34c4eb0d90d11ac53abaf38f8fef09fb3dcc55b8453718aa73f39508f6c54c9	teslacrypt
509.	b3a850d2572eaab768fad7c1a603206a429efdd56527fe44deca9d68c8956eb2	teslacrypt
510.	b3b18d81bece3b45f5a7210c893eece5e902c8ac3c28ac0d1ce8914b0ecb2196	teslacrypt
511.	b3e51004c582d0d1744dfcdbbbce94b60e75ff59392211d4b7b83eab198905a3	teslacrypt
512.	b3eaefc90eb5a958dd8180b2c4eaa708b6657fc259c52401e3b5ed922dcd712b	teslacrypt
513.	b42727b961791f14c8a67eac81abc030fbd57d88aec126e7da6893628e0637a3	teslacrypt
514.	b44e515413655d68ca5faa1d578cc52c295ff1db5c379ba8ba95728360003441	teslacrypt
515.	b45d6ca7a9cc35f9e037a5789fcb7aabc4d8454f8ccb944c17d8edd9d8668c36	teslacrypt
516.	b472cbcee12ac177dfce839fc95f4e6bb9ab21cd9e0ae71ca7980c01957bda26	teslacrypt
517.	b48aa32f96ff0d14316c2a7aa8e14864fdfdcb6a29eeefa132b2bdcf8f882401	teslacrypt
518.	b4e9081a8b072d3e27c6d6e67daf741c986abbb3197d2517f81ec17904e38af3	teslacrypt
519.	b51827b8e18e07ed456c06b73b9fa4a366838f51b77407079b333a8463748abe	teslacrypt
520.	b531534e77de7902aeb9f3c4bd624b98d94bbf34607157f7bfc99370446e7f71	teslacrypt
521.	b5444a872ceb5c670452093ea15d32f4afd4a33eea8416a5a51aa1870992e676	teslacrypt
522.	b55b3fb2231704d19ffe331d833b5c9606a56095718b0a89e65cf70c5f8b75d5	teslacrypt
523.	b5d927a9ddb3ce823c5d907c3790ce7afd8ba4cf6adf42cba1055390682e7e6e	teslacrypt
524.	b5e22875ece6a9b4d5b12960692031fa1e4c9c4a8e52fdb3a02acfd9ea5e44a9	teslacrypt
525.	b5eaeac7e3f6fe4c4bb5a691dd493c8931b421cff45b49e682e5510b94a5182e	teslacrypt
526.	b616ab4d5e04d670230bcf28711c7bd0d40d1a953064a5c52a6bf7d6b370ffa0	teslacrypt
527.	b677f845323a8495cbf34dda3dcc2b22d754e48daedab1a57f9ee60660f6b0cb	teslacrypt
528.	b6be17e9df1d00d9a456789ecc5d98d6b3449fce0ab28b0afcf2440602931215	teslacrypt
529.	b7331f6a8d409369e4855da5a743993d441d474e1cf5f8577f310de0934b947b	teslacrypt
530.	b76de9d7dece1ce3dbb44adbc303784900f5ee6836575e287837a55fb22a6f54	teslacrypt
531.	b780a9af30b0c38cd06d4a2fc89acfdb5e11e7da6596c5be7fb6c5884d9ec652	teslacrypt
532.	b805d07c0655f190f1ae17255bacf2eda112c0e3b736b82c81b104700d025218	teslacrypt
533.	b8330680479cd7199263b1d43377f007cb8a2a97d8a9e4722184119f2ecda2dc	teslacrypt
534.	b852781323308046a5122ad660f2241fbe898184412e2211f691a74e083ba353	teslacrypt
535.	b89d721689347baffae66d1e186f654eeeb17d69e82b3a17880587b53aff1067	teslacrypt
536.	b8aa0c05a23aa1acb4cfa88c8b4a942d89c0f4ecb96d01792cd4e53733d21c78	teslacrypt
537.	b8bbd68a4bf88a6480cf621c100a632a6b53f637208c8bc06dc81e8d2c6f227a	teslacrypt
538.	b8c618753fc84b9d09111c1f231e7befc6e196a2276fd2a3d25041e094e589cf	teslacrypt
539.	b8cc81afaf8ac54aae037ccfea44d7de8143de9ad321e9ca06f26e2ee5abfe4f	teslacrypt
540.	b8da005b62db3d349ab4cdb778946159ddc7920cc25f454f174465e9172c03d5	teslacrypt
541.	b8e98d1fd1932adf2cbd689f60ee5eedb90df71b23987dda5efad99437c42116	teslacrypt
542.	b8eb96cb9b812fec4b7c6b0d5e6eb7972742c3be9215295ee1cb337ab595ecff	teslacrypt
543.	b8f09d9c2a8d1b642f8cf353f6ec637e2932ea63e7d3bb93c67cc95c6c340570	teslacrypt
544.	b8f96bc6fc06940c8895681fbd19baf701ad7abba85b95af81404c6fe37ec91b	teslacrypt
545.	b92fda23d7f6dda7c06d9d95f7255650cd94bd68ffa0ad1566ba0d3ef97d2a18	teslacrypt
546.	b9535c1e49dd2cc3a12c74189835d9e1bac0fa8769b0554cfb8a84aaf8d63a68	teslacrypt
547.	b9588bb8cea25df37e97e1f587044b39906b31fba93e33722068df22737ba14f	teslacrypt
548.	b9738971d60481d70ff1fbd94727a8c564926f9cd85b40439ca7b45043b87f4d	teslacrypt
549.	b97c3af5da547f58406cec9f61bee5d4550966fc816a17d551c8463b10b7bc90	teslacrypt
550.	ba0f4354c07192e45ebedfc42113beb1c2dbc7f47e7c41e90ca820d12e4728e2	teslacrypt
551.	ba2cafaf1ca41aaf8f9100900f4a31548d1b870de453521f671f57b48e1f06e3	teslacrypt
552.	ba3fe4abc9a06d4a9c68f4195f5f5670b96258b84ddae4d166334692c0b88804	teslacrypt
553.	ba42a56260689e5ddec3941b8d2d466130428f3b5ba84fb3bb506f8d79a4949e	teslacrypt
554.	ba90714b1aea320b270b8d426db711563dd290058c9589463f450a63c01f0150	teslacrypt
555.	bb04ad35dfc02aaaee97d64ad3d91e33a087b4de86e48bad5063dc01cd38f031	teslacrypt
556.	bb05778794d3a117f9760873f0651a28dfb27ffd63b2b3c5a8024e3b50ada8c3	teslacrypt

557.	bb1b3ce59b12f81311380956af42abf62c35f791675254680c706fe3f3fe1cb7	teslacrypt
558.	bb211162410b307fb696c948f3627c093c194615df0be0f4568ed30f02f9703a	teslacrypt
559.	bb2f0977d817b4e45c39b54594776a3262bc1712174e83c679a0fd193c93ba00	teslacrypt
560.	bb591ef4f2ddbf105702921445a66fc3bde111aac0130f7d2bd8f1b23b49baa1	teslacrypt
561.	bb6dadbd7a10da2ae19f9ff4c523852f1d2c39e168df0d512962ca188da94971	teslacrypt
562.	bb6f9ad934c29228c8545eeece0a861a2d5fb3910e357e58b092f1c9d4399785	teslacrypt
563.	bb75d79e347c780e44e1a63019ff2c90e9187f81cbcf5524f4a7e5b4d69dbc27	teslacrypt
564.	bb7d747a4035161c23f7d1e09f5f0a7f4adefe34464ae6dc364a4da77c609a02	teslacrypt
565.	bb7e7916cc971fa65fb7bb60601406f7ee44e320d5952282f1ab24a55a6e5647	teslacrypt
566.	bb8358b0ade54709167ac78cff5202099108316ff50f429d9c0e111cfd582e22	teslacrypt
567.	bbb6334d2105694a7f191109e1176a329915e0a8587be2fed74855f757b5331e	teslacrypt
568.	bbf4ecfe09582ab6e4e5556da54c656340aac17f0bd7ccf1d179c87586665e78	teslacrypt
569.	bbf504203274ddbbb2ea834a71a61989f899d160b3822de6bcd4839a0437663d	teslacrypt
570.	bbf57eb7249a998bf25c84f2c04dd4de213de6d2e585eea5a3c4d0c677e5fd68	teslacrypt
571.	bc03d234914b252ff7535fb7fea8b2ed22237338216c81a6cb8caaabba18bff3	teslacrypt
572.	bc1bfe12092b60a564c8eeda9dc1fb4bcf0639bbd58973e29abe6ed11da39a11	teslacrypt
573.	bc39bc6e08e20cd9c294f0bdb7c29cc9a8c42be1fcd38c3373a80d7afb0cefe3	teslacrypt
574.	bc56e20a12632ac231e94ffadd4c1575713c55dbb9e23ff62e971f7095542022	teslacrypt
575.	bc69fe76aab079814d8c2e8ccce46050bd9ce50e0ccd5a549b9bef7958c3b88a	teslacrypt
576.	bc9b2eb0e3327d45905fd1166ad32c736959ad51b1d9c0a4f56b97cf1be2b575	teslacrypt
577.	bcad0e1e7308b6a658a45a88480f25b3bf363a72badb1c96d30718a36fb76015	teslacrypt
578.	bcb1600f39a35f1ee217401c2a882f06d1f304dcfb95bb6d2f0e59a673c34e9e	teslacrypt
579.	bcbcc35ece2a1f2ffc045c3b0bea20778c50c0af3689ad71f6305313f6ab57d6	teslacrypt
580.	bcc87097c8d497223e5b8d5218b9aaf9426ac019f75d19affd7aacade331037f	teslacrypt
581.	bd13b61234e350683c080b4e9c3654ad83c3c55edc1f733d8e0cc8d477da0406	teslacrypt
582.	bd285ff14886cea17170a5e21843cee32ea05b625fac628637b2677a250554ad	teslacrypt
583.	bd32b246b625887ce3818594950dc3e0046a3cc3e8d9b2b5dcd1e81d984f169b	teslacrypt
584.	bd45cd253430dd0f90e85a8eb5249739142d0fa3521e46a150122fdc25354788	teslacrypt
585.	bd787382a63967f2f7e3d5a0f6ab09bac660449bfe5bbf713ea394cc350cd56a	teslacrypt
586.	bd7c5069c503d4e3a270fb7baab7b86470fb34399a659db12752902be3ce8e04	teslacrypt
587.	bdc338d3ce1da1425e4a69962c692316199eabda5aa691bfb86efd32a6a997d4	teslacrypt
588.	bdc6db80c8740d8e8ff9c650dc6ce7de30843eca86357b2a9159da423c2b651f	teslacrypt
589.	bdce972d0fd1c293a5e40cdb29f38eb4295bad843ddb33c1215877f780a70be7	teslacrypt
590.	bdde3b0e3bff437f6aa806119a8224900980d658300b42c649c587ba569fb659	teslacrypt
591.	bdf1898880dcfb8b4a797f8df719f956b1ae42c1229c62821e6cb4e3d9ca16f9	teslacrypt
592.	bdf47e8f094638be5788d0dad0cb98ca358b4041063a277a0880322107d477d4	teslacrypt
593.	be21d5f6cb4e6d4f86145a7db8dfdcc4dac98188a9683ed2ebd8e9e81ebbb684	teslacrypt
594.	be3f95753e85714d042471cb2d4fe9f54fb6da6fcee3815cd9093af6792eba83	teslacrypt
595.	be4704dab2034d6fc17d90d9fa8161be9ff0d88c31084b793b302ffada9f9153	teslacrypt
596.	be543dbb798df3e3bdfe68b08c6e36bb592c9199f9c25a5d5f37ace1dc4a2790	teslacrypt
597.	beb645a5d859bd194e960ca158f60488f0b42ea739841559d4593c3bba226501	teslacrypt
598.	bebf2eb0fd5b95350f205aa96df341c0b148c66f225a7b6a4ddc06d7911e3d2a	teslacrypt
599.	bef9e3a479450aa6b6b9421f6c28c93bddb45a54cf0cef89a6bcc31b6d2e4a4b	teslacrypt
600.	bf24f79c058537238a3c2c624a495d3871badf1b3186b10c74da26cfd25f2b83	teslacrypt
601.	bf2b9e837613d15014e8086133b513aecaf3669b7f9e21903a53e977585b5778	teslacrypt
602.	bf36864ee75ce062994b3c67c47aad4fd09a64c75c6118b5ed66bed5f6a11ab0	teslacrypt
603.	bf3ffb582b639abc9350aad1a19996ff56f42e0207fcb658b9f268e00a77030f	teslacrypt
604.	bfab2ed9a9e9b6d6d064f11c30f4ddce9dfa3c029b6ad3ecdbdefb18e7d7e91f	teslacrypt
605.	bfbe024dcdcd2f4ab896d3e48d267547bbc0a8e9948ac50d05a41f2c4fdba980	teslacrypt
606.	bfd8ab321678e37c1e3c5236d70a0836f063e76788a7683da12c2989ca7a69d1	teslacrypt
607.	bfd9843f266e28caa90632742523045f59c21d100d582e89c0a2d402e30f4f10	teslacrypt
608.	c07dd40ae4006ef7027ab788110dafe06bff0eb87f864151f34dab1f10b730b2	teslacrypt
609.	c0b06d5ec84d054dc925b1c53da7371eb5b033501be5188d37169f416e9db738	teslacrypt
610.	c0e9f8bb5079291a5947bc2d573adc4ef684144463c247f771c8829320662124	teslacrypt
611.	c0eeb26d4900ff586c830bd2e7e0f9174783f36c1dbe8f43c6d090133bbb5403	teslacrypt
612.	c1081d1cb2f607b98e8dabcb6e3a4baef1c42804c8334dbfe0cf7e5475d5425d	teslacrypt
613.	c10f5d08c6ed8a614e2b15a3b71f0e1627b69cccb68394f250da2f3b2dd74240	teslacrypt
614.	c1669a4409d40079f4c7ac3e89e44fbe25823714ed05e096d6b5a3ffa15b2e43	teslacrypt
615.	c1684484c4f41dd18ce2654e991a46cb7af6242d82a518f7b6b72227c0d6aa3b	teslacrypt
616.	c17462ba0f8ac86da8c045aedabadfa7a6fbb30d05d9c01a13f5e1f5cf439a8f	teslacrypt
617.	c184657d45e3474ff6d88937628a3fd15c6b0ebae4e6a4431a6c46047a97d633	teslacrypt
618.	c1871cefd7ee74c36f5d31103d6217e7a11acde91db7f7ad761b4218254031dd	teslacrypt
619.	c1d36e3cc137c96e6ad0378e22c9e0d46f67848c61ee41dee9eda8719eb67fa9	teslacrypt
620.	c1eb32bf68716bfff4524ad31198b3469897338421cac4bc6d2e0d38dcaed9be	teslacrypt
621.	c219ee4f45bdcabcc3c32592904387376c47600b4a67a53661a210c6fb6417a9	teslacrypt
622.	c22df6e701ac6b450b72388244355d4ff2cc77961ad53456eba6e72483e031d6	teslacrypt
623.	c2312ae4eecb67956ee90c8af4c6c5f11250229749e8cdf0625070a4f8e52381	teslacrypt
624.	c2338ec0e9c93d48098225c011eb0b5b9f587df5d37bcde710841c96b526147b	teslacrypt
625.	c2a7b17a5f0ea6acfa039a16233286b9e170a46c16b01d96a9d5c99233c07b79	teslacrypt
626.	c33ff3581378fc2e2144bbc9e9f70c95c17552d673d5f8c677f53da25fad2661	teslacrypt
627.	c341663ebbc9a17a763429579add0856e87b677b46168c7d6116e102a538c9cb	teslacrypt
628.	c343010972b7288f5baa33e9b18213216b008cb232e45e3edaa20f1d59131b4f	teslacrypt
629.	c34a5a6505d8ed20f5f035ba4bc9cdf0aa248731d5e3487142b8d4e068d3985c	teslacrypt
630.	c354172152c9d4fb685a007edabda9f7956ce18587f9823bfa35eaf5a3d0f746	teslacrypt

631.	b920a85be678d319e5ba7f8cf03d355ea16f0d312541548308e904bd98faf1c5	tpyn
632.	a157552da52aabb588358d955beb408d20f591850464953fe17e3324159c3e2f	upatre
633.	a9ea3e2037b3c0a60c7c70f9dc41ab71d6edb31c99f67abd972e589f8a353cb8	upatre
634.	acbf87a0794e741661ab350436f392fb6e7fb2ded4b073b30fbebe482590f9f7	upatre
635.	aede6188320969700b47a266f2f60431a2050daf4e32752e810019289126870b	upatre
636.	b78e9706de87a891148c326f0afb7ba18272d8bf0d72b7a8b977ef7d80d73971	upatre
637.	bb68d82cde5e245d7286e5645e84e6631f9b47a0e2d0195dc04fadf7f06cf28d	upatre
638.	c2245255acf187f369a55a00faf60761688a6a6e8a256b1a1dc1e56360fa04d1	upatre
639.	ab10ac6549a773a1159223aa46aca4f3c3a9415cd1cc9ced3a206ec22e671579	virlock
640.	b05b748948190eb62591dbcd0f81622113966846c421192b7aea130361730204	virlock
641.	be4659558111d73bc452dd8bc7bd3df19181c6f979af68c235d9ac3c97ec832d	virlock
642.	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa	wannacry
643.	0dfec219787f68baf2a597ea975272505337b1501515eaee1b0611a904155ae7	wowlik
644.	b69a176ee25e2d35b80ded627b751f77d56cf2ad27626cb8bbd3333d78284fca	wyhymyz
645.	0b2c7a4498da4763033d464ff3ae66f9d4fdfef1c618e70e93f08b4107891401	yakes
646.	0b5543d5e29841f93225ff5df5660ec65a254aecbd68563be9c42195bcd65a1d	yakes
647.	0b980f26d4288d9ea39935e8ef9fd320f963c5edf69e863678d4981bcb9614b8	yakes
648.	0cd4d6f564f8a413742b5c393ffe837be39c803bda9de58490aced9b6aa9132c	yakes
649.	0d416a7d6d0dfba59728f6c3382b93f3e293c143fde35d864e99a7fffc86a764	yakes
650.	0f9a3a7daf103c0ebff3b025a26c61aac9e281193b646cc9e5f8957f42380fff	yakes
651.	0fc7db597700c57db258d51cdafd61c9b069563bb4e5a1a65bf2cc202e16bc2f	yakes
652.	a076c68f138fded6cfd873298a7ffd989eba8a5a099569d2e6f57f80586ebba0	yakes
653.	a0f537ecf496738ef50afd1ae2aa9ef7c3eb86b166680c18f3ffafccf507a035	yakes
654.	a16672a3a8667cc78580bb88a2846febebfaf4d17dfd92263534c6fc359821ad	yakes
655.	a26055d6703af685ca2c02d50698098f4f921704a30482152545c1cc9ef30069	yakes
656.	a3633a6f3c7e1231b9e3dae65290415bcde4e7d502479360674fa82f084e206f	yakes
657.	a3b76ea4482ec4daae21cc1f5e682a972e29280781b705667ce564ad00a0154f	yakes
658.	a424cc306674236588eab8f8ee0bdad4843f4ba8da2052476a99e46a05f2a238	yakes
659.	a677620dfe3676d80cbb655e5a28eab477256a1023418c225002e6cd2e886e3f	yakes
660.	a6e9b7434b50e0c780e0dce770c9481de055fcffe27e3425477de0b8d3e8d97f	yakes
661.	a8c2b64c358bd67adebd9e2f17d501a4a93881579248a0c23623caddd25f137c	yakes
662.	a920a6ba7f0527d47bccaa19f160781f94b7aee1770f12325ba19a804632f919	yakes
663.	ab655adbaeb93479cd8eb04033ea50653aeea2fa1b9d4c5489cf539fe8552cb5	yakes
664.	acad5aba77bfa58f8070a9021193676a4bc7c2ef2e3359d25ec9d19882dcf04a	yakes
665.	b185a417c6f4fb280c3c6d76ca7cf5bd7e28e31f1dfa2c8f5da0a40667469fbf	yakes
666.	b1a65248fc29247553ae6e518076d9bdd75bda9d249e126c2b161945138056f1	yakes
667.	b4404ef88c7d181feea484ccb193c9f4fa1337c1a0dee1b4f44a9686bf28b702	yakes
668.	b5ef5c737057eabca8945e1938b892336a313a89a3b11970139b2fd2e9dd3625	yakes
669.	b61828b84dbd1d535d1891804978826153341bcf75b1ba78c601451d70df28ce	yakes
670.	b846568e67669b55964b8199a30c67810b15ae8450bdd9af669a5f67784b2088	yakes
671.	b85fe2e1ab522683bd497323879997bf5862501244b6ba8f74be0d1a38860c99	yakes
672.	b8d0aa06fc1f7d78280880e88ce2f564b527ae739e5b11bdf1affc15118124a1	yakes
673.	b924dcdfeb68ffe8fcde35e1e22cf44a77c8b1bad9b0d575c016816fe5cef471	yakes
674.	ba0a7fe13f590f48d965849c71e6c367b58a99a9c7ac2cb23cc3a39ab4548917	yakes
675.	baade55821754da5d83c93bb412762d985a846a24de683b8faab7e38bb411676	yakes
676.	bbd8a4b1d60547638c67c325a3ce1b449758146c3e3a49731c3ddaf8f987c0d3	yakes
677.	bc5ed05773aa9debda689eed2f8e347b8b525dce4e1d1de7aef095db3192663f	yakes
678.	bcf1244bd09edbe491da03912e6461f13730d34b5974443ed97a1ab06bc94f3b	yakes
679.	bf564117f1c16a15ee1fe5bc06b6d7864224c0a8e27a148acbe90f43f509ca43	yakes
680.	c101ab4c9d2558b4c992a12754e7563cec3a8762869ff9ecc24b52bed1afa9c0	yakes
681.	c2d4f240122354a6db5edc428dba1b6d6540dc10818d5561128690d57388c508	yakes
682.	00e0302c80c817dc4cd99c2b30012d74a3fb2eecb30c9fb0ab156bcec26b9b30	zerber
683.	00e437c217873dba5baf242f4148d05ebc7beb613879a1b2fb366ff03e653098	zerber
684.	0ad4fd25a9611201d3125cdf129bb3ca3738cc49bf3e3bfd53f6a37dc6aa6768	zerber
685.	0af56173b6a8d920e8f42c564d590373d8a8c55edda2476deff5013a39d76d87	zerber
686.	0b17efb0e0348fce60a5c6a46ce247f45b9f6593717338c3468b43f4b406bd92	zerber
687.	0b272278a07cf126424f743706929244b1ea9b2e4dc5aada70de23f970d109ae	zerber
688.	0ba9ea9d04de8c5235da83f5801ed9d265b38f634af199f62c25fdb275bc32d3	zerber
689.	0dc1387d188ec53b85099ec7efae0b353836f6963972c24d4a257648b0b20d8d	zerber
690.	a120b7cfdf3909fa9c8645ab377f30ffdc1e65f65f945190c491286e99b0f6fe	zerber
691.	a1cdc2a8037b153add1aeffa52501f3fa48e05f2d36921bf559b080844428ddc	zerber
692.	a2d0530dde8c6cb0980fa7a69aec3cf2e355c121149cb2f08ffc727bfcbec53e	zerber
693.	a2e49c142083facea31250787f41fedf889519a202376e0519cbbf9fdf22299b	zerber
694.	a3a082e22028228f0d1fadb58e9d6bec435094d73636b9af8a4b8a8c08617b37	zerber
695.	a85d8fce5d2f587580773c2c4d757a112cc775e4581902f9467aad70477c0a86	zerber
696.	aa03e03ff2c4e3fc90864ab9b859bb575adabee458d29fc1bc4673a02c68518e	zerber
697.	ac81016b4bb9755c249bcf51a9087f0099c8240df7c852158a18f23914c05df4	zerber
698.	ac86648cad6745408e9f2630cebedd5adbd871b603641c0f03cc0f2af5de5ed5	zerber
699.	ad7b25abcaeafbedef2dfefa5fe4cb49896f3e47aea7ecf5508f67d48fbe1471	zerber
700.	adf93fd3bed49ed68ca06bf22b21eecaf3d712a0f1f7c7d46abd2c667d406c65	zerber
701.	ae53c4359f5468482f07f1cf3d1f12b4b736278e3bf8e66425a9a325459703f7	zerber
702.	afff4fb5e707667c7bc7b4a94df413300b0c8332219608f54f4fc0df0b309140	zerber
703.	b160f541cd35b23b9675ce11b63c32b453a5e9636dfef0528b88cf33f3870fb1	zerber
704.	b1a5cab3643deb848d31b94544177102a681f80a37306c4434f6248eddb3f5e2	zerber

705.	b37a24b3b81713df91dba5130dc2ace35a5695289c52fa75e3a5369e82df321b	zerber
706.	b382d8ebcd2a775e51cfb846095824868dad1d32b4f7498b1672a7e39880da1f	zerber
707.	b4283ecafe0aff5b3b5641a8aff6859cfa4f5b56f6c261c39325da8c5d4e3f2f	zerber
708.	b5109889f41d66bfcb3336d559fbd52a76da588a8650597073c7cc7441618e8d	zerber
709.	b512a3e267ca0ac1ee2632a6cf0dd258147a4d1ddebeb7f187974f1ffb9f53fb	zerber
710.	b5910ab542bf639d161aadf7f863b563df80a2bb964c910c14303bcaafc5fb00	zerber
711.	b70cba09041556562dc68f9396a62f346853570f9837c78a4bc72fcb6f3d2c5f	zerber
712.	b737063ee1cdbbf188085af8635088a9bc27fbad8f1eb5ee2a69bbf7a8be916f	zerber
713.	b764cbee3765313309efeeeeba29f63a02c60b6dd51bc977d564e60516d96db7	zerber
714.	b8487e436c78d18b66bf580c150615837c7d66c3a950a3cd5258d9ff936162a0	zerber
715.	b89f67e3c5f343e4ab953e1c556c1daee6e5eb04afef5d400859016c134c0723	zerber
716.	b8c092dc8589d684857401e7053ce8f773639bbad158499de4cbaa75f7329c13	zerber
717.	bad912d0798b4ae5f7bdd4424310cfc40fdfb6d8752b3ae346ff3db2f9627870	zerber
718.	bcb4e6b2d7ba39c39315b0106d3011e197d2b9aa0f5458f89c5db2c1b5da24c6	zerber
719.	bd0d0db1bddc1ddfc9aa38611ed4d94976dcbdd18ef4b15bf8f3ad323bcfae1b	zerber
720.	be86d51ddc52ec55e30a170bfde4584655d5f3a9ac56688c7c49b99352e7842d	zerber
721.	bf60c5fd440899d8b99c209db47a9f39a7ca31d0bed7c831b1c5f23f533abdde	zerber
722.	c01ead04d9a4aabeca56feba7905b64e7a6c2b924cbfe38ac25806143cf668a5	zerber
723.	c03a4a0412a9c8c50f73c0f4020fa035456830dd676cdcc7ac2d0214d324f109	zerber
724.	c0a91edf65cbaddb99c2d9dda1210b62d0f0895edf49dc44eb1618e9f81ee1c3	zerber
725.	c0ed6360a5176ecc284953e231c0c1b262b429e0d74836bdb0de5166f7be981f	zerber
726.	c123b58199f6b92951d27387040cad294f7ab457fdddd529760925414827ea89	zerber
727.	c1448598e2afcbd8adda5d6e3f24a8609552cad4ad9a25849b56cc534cabcc9e	zerber
728.	c1fcab4d6d19cc97f30e8cd0d8c7fa8ed6fe206f8160082a1b22594a86362067	zerber
729.	c2e7ccd6077c043ba722b3ee141c95cc10dd506ce26ac18be29c8ef8ac26971a	zerber
730.	c320a7c9282551177083af0fefdcb47ceb3e721a665314eea759099de82bf89d	zerber
731.	a261d7a919495ac349c125dadf89cc557352a7d2feedf3898fa5e320c4c75efa	zusy
732.	a6496f696b3792fffb4cf1b009b6ce76a39a3f410cd2692b71a55474746c4368	zusy
733.	b00ce6a6107f0ace12878c3636fd42494bc387ba494f87c094c3597eb1dd4943	zusy
734.	b9fbc8deb897ea739ae10986e0c16714d5468e7b960ca4a1c93fadd39fb54486	zusy
735.	ba99ac9b25cf0b78872b502cc1ae43df86cce96ff2a22c378ff17d7c03497b99	zusy
736.	c10eeb0df592f972d478bd4f5de86a092df17060837e516503fa964d837287c0	zusy

Bibliography

[1] Xenos: Windows dll injector, 2019. URL `https://github.com/DarthTon/Xenos`.

[2] C. Brunau. Datto's global state of the channel ransomware report 2018, Nov 2018. URL `https://www.datto.com/blog/dattos-global-state-of-the-channel-ransomware-report-2018`.

[3] K. Cabaj and W. Mazurczyk. Using software-defined networking for ransomware mitigation: The case of cryptowall. *Netwrk. Mag. of Global Internetwkg.*, 30(6): 14–20, Nov. 2016.

[4] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi. Shieldfs: A self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ACSAC '16, pages 336–347, New York, NY, USA, Dec. 2016. ACM.

[5] Cuckoo Sandbox. Cuckoo sandbox – automated malware analysis, 2019. URL `http://www.cuckoosandbox.org/`.

[6] G. Cusack, O. Michel, and E. Keller. Machine learning-based detection of ransomware using sdn. In *Proc. of the 2018 ACM Int. Workshop on Security in SND & NFV*, pages 1–6, New York, NY, USA, Mar. 2018. ACM.

[7] Cyber Threat Alliance. Lucrative ransomware attacks:analysis of thecryptowall version 3 threat, Feb. 2015. URL `https://www.cyberthreatalliance.org/wp-content/uploads/2018/02/cryptowall-report.pdf`.

[8] M. Daniel. Cyber threat alliance uncovers cryptowall version 4, Sept. 2016. URL `https://www.cyberthreatalliance.org/analysis/cyber-threat-alliance-uncovers-cryptowall-version-4/`.

[9] B. Gammons. 4 surprising backup failure statistics that justify additional protection, Jan. 2017. URL `https://blog.barkly.com/backup-failure-statistics`.

[10] A. Gazet. Comparative analysis of various ransomware virii. *Journal in Computer Virology*, 6(1):77–90, Feb 2010.

[11] Z. A. Genç, G. Lenzini, and P. Y. Ryan. No random, no ransom: A key to stop cryptographic ransomware. In *Proceedings of the 2018 Detection of Intrusions and Malware, and Vulnerability Assessment*, Cham, June 2018. Springer-Verlag.

[12] Z. A. Genç, G. Lenzini, and P. Y. A. Ryan. Next generation cryptographic ransomware. In *Secure IT Systems*, pages 385–401, Cham, Nov. 2018. Springer International Publishing.

[13] J. Huang, J. Xu, X. Xing, P. Liu, and M. K. Qureshi. Flashguard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2231–2244. ACM, Oct. 2017.

[14] G. Hunt and D. Brubacher. Detours: Binary interception of win32 functions. In *Proceedings of the 3rd Conference on USENIX Windows NT Symposium - Volume 3*, WINSYM'99, pages 14–14, Berkeley, CA, USA, July 1999. USENIX Association.

[15] V. Kamluk. New gpcode – mostly hot air, Aug. 2008. URL `https://securelist.com/new-gpcode-mostly-hot-air/30446/`.

[16] V. Kamluk. Another way of restoring files after a gpcode attack, June 2008. URL `https://securelist.com/another-way-of-restoring-files-after-a-gpcode-attack/30434/`.

[17] V. Kamluk. Restoring files attacked by gpcode.ak, June 2008. URL `https://securelist.com/restoring-files-attacked-by-gpcode-ak/30428/`.

[18] V. Kamluk. Gpcode-like ransomware is back, Nov. 2010. URL `https://securelist.com/gpcode-like-ransomware-is-back/29633/`.

[19] A. Kharraz and E. Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *Research in Attacks, Intrusions, and Defenses*, pages 98–119, Oct. 2017.

[20] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium*, pages 757–772, Austin, TX, Feb. 2016. USENIX Association.

[21] H. Kim, D. Yoo, J.-S. Kang, and Y. Yeom. Dynamic ransomware protection using deterministic random bit generator. In *2017 IEEE Conference on Application, Information and Network Security (AINS)*, pages 64–68, Nov. 2017.

[22] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele. Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on*

Computer and Communications Security, ASIA CCS '17, pages 599–611, New York, NY, USA, Apr. 2017. ACM.

[23] B. Lee. Ransomware: Unlocking the lucrative criminal business model, Apr. 2018. URL `https://www.paloaltonetworks.com/apps/pan/public/downloadResource?pagePath=/content/pan/en_US/resources/research/ransomware-report`.

[24] K. Lee, I. Oh, and K. Yim. Ransomware-prevention technique using key backup. In J. J. Jung and P. Kim, editors, *Big Data Technologies and Applications*, pages 105–114. Springer International Publishing, June 2017.

[25] W. J. Liu. Process hacker, Mar. 2016. URL `https://processhacker.sourceforge.io/`.

[26] McAfee Labs. Mcafee labs threat advisory: Ransom cryptowall, June 2018. URL `https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/25000/PD25480/en_US/McAfee_Labs_Threat_Advisory-Ransom_Cryptowall.pdf`.

[27] S. Mehnaz, A. Mudgerikar, and E. Bertino. Rwguard: A real-time detection system against cryptographic ransomware. In *Research in Attacks, Intrusions, and Defenses*, pages 114–136, Cham, Sept. 2018. Springer.

[28] S. P. Miller. Dependency walker, Oct. 2006. URL `http://www.dependencywalker.com/`.

[29] S. Morgan. Global ransomware damage costs predicted to exceed $8 billion in 2018, June 2018. URL `https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-exceed-8-billion-in-2018/`.

[30] L. H. Newman. The worst cybersecurity breaches of 2018 so far, Sept. 2018. URL `https://www.wired.com/story/2018-worst-hacks-so-far/`.

[31] A. Ortega. Paranoid fish, 2019. URL `https://github.com/a0rtega/pafish`.

[32] A. Palisse, H. Le Bouder, J.-L. Lanet, C. Le Guernic, and A. Legay. Ransomware and the Legacy Crypto API. In *The 11th International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 11–28. Springer, Sept. 2016.

[33] A. Palisse, A. Durand, H. Le Bouder, C. Le Guernic, and J.-L. Lanet. Data aware defense (dad): Towards a generic and practical ransomware countermeasure. In *Secure IT Systems*, pages 192–208, Cham, Nov. 2017. Springer International Publishing.

[34] R. Richardson and M. M. North. Ransomware: Evolution, mitigation and prevention. *International Management Review*, 13(1):10, 2017.

[35] K. Savage, P. Coogan, and H. Lau. The evolution of ransomware. Technical report, Symantec, Aug. 2015. URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf`.

[36] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler. Cryptolock (and drop it): Stopping ransomware attacks on user data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 303–312, June 2016.

[37] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, Sept. 2016.

[38] StatCounter. Desktop operating system market share worldwide, 2019. URL `http://gs.statcounter.com/os-market-share/desktop/worldwide`.

[39] StatCounter. Desktop windows version market share worldwide, 2019. URL `http://gs.statcounter.com/windows-version-market-share/desktop/worldwide`.

[40] Symantec Corporation. Symantec security center – trojan.cryzip, Mar. 2006. URL `https://www.symantec.com/security-center/writeup/2006-031314-5208-99/`.

[41] Symantec Corporation. Security response: Cryptodefense, the cryptolocker imitator, makes over $34,000 in one month, Mar. 2014. URL `https://www.symantec.com/connect/blogs/cryptodefense-cryptolocker-imitator-makes-over-34000-one-month`.

[42] Symantec Corporation. Internet security threat report. Technical report, Aug. 2017. URL `https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-ransomware-2017-en.pdf`.

[43] Symantec Corporation. Internet security threat report. Technical report, Feb. 2019. URL `https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf`.

[44] The Barkly Team. Must-know ransomware statistics 2018, Aug. 2018. URL `https://blog.barkly.com/ransomware-statistics-2018`.

[45] ViruSign. Virusign, 2019. URL `https://www.virusign.com`.

[46] VirusTotal. Virustotal, 2019. URL `https://www.virustotal.com`.

[47] WatchPoint Data. Cryptostopper, 2019. URL `https://www.watchpointdata.com/cryptostopper`.

[48] J. C. Wong and O. Solon. Massive ransomware cyber-attack hits nearly 100 countries around the world, May 2017. URL `https://www.theguardian.com/technology/2017/may/12/global-cyber-attack-ransomware-nsa-uk-nhs`.

[49] I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *2010 International conference on broadband, wireless computing, communication and applications*, pages 297–300. IEEE, Nov. 2010.