Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks

Amin Kharraz[1], William Robertson[1],
Davide Balzarotti[2], Leyla Bilge[3], and Engin Kirda[1]

[1] Northeastern University, Boston, USA
[2] Institut Eurecom, Sophia Antipolis, France
[3] Symantec Research Labs, Sophia Antipolis, France

Abstract. In this paper, we present the results of a long-term study of ransomware attacks that have been observed in the wild between 2006 and 2014. We also provide a holistic view on how ransomware attacks have evolved during this period by analyzing 1,359 samples that belong to 15 different ransomware families. Our results show that, despite a continuous improvement in the encryption, deletion, and communication techniques in the main ransomware families, the number of families with sophisticated destructive capabilities remains quite small. In fact, our analysis reveals that in a large number of samples, the malware simply locks the victim's computer desktop or attempts to encrypt or delete the victim's files using only superficial techniques. Our analysis also suggests that stopping advanced ransomware attacks is not as complex as it has been previously reported. For example, we show that by monitoring abnormal file system activity, it is possible to design a practical defense system that could stop a large number of ransomware attacks, even those using sophisticated encryption capabilities. A close examination on the file system activities of multiple ransomware samples suggests that by looking at I/O requests and protecting Master File Table (MFT) in the NTFS file system, it is possible to detect and prevent a significant number of zero-day ransomware attacks.
Keywords: Malware, Ransomware, Malicious Activities, Underground Economy, Bitcoin

1 Introduction

Over the past few years, a class of malware known as scareware has become popular among cybercriminals. This malware takes advantage of people's fear of revealing their private information, losing their critical data, or facing irreversible hardware damage. In particular, this paper focuses on ransomware, a particular class of scareware that locks the victims' computers until they make a payment to re-gain access to their data.

Although the first version of ransomware appeared in the wild almost 10 years ago, the volume of ransomware incidents was not significant until a couple of years ago. As number of ransomware attacks increased over 500% on 2013 compared to the previous years, the ransomware threat made the headlines as the most notable malware trend after targeted attacks in 2013 [37]. For example, the `Cryptolocker` ransomware alone managed to infect approximately 250 thousand computers around the world, including an entire police department that needed to pay a ransom to decrypt their documents [15,30].

Given the significant growth of ransomware attacks [37], it is very important to develop a protection technique against this type of malware. However, designing effective defense mechanisms is not practically possible without having an insightful understanding of these attacks. Currently, many of the recent security reports about ransomware [12,15] rely on ad-hoc procedures rather than a scientific assessment. Moreover, these reports mainly focus on the advancements in ransomware attacks and their levels of sophistication, rather than providing some insights about effective defense techniques that should be adopted against this threat. In this paper, we investigate the key functionalities of ransomware samples such that we can propose effective detection mechanisms leveraging our findings.

We created a collection of ransomware samples that were categorized in 15 different families. Our data set covers the majority of the existing ransomware families that have been observed in the wild between 2006 and 2014. The data set is created using multiple sources including manual and automatic crawling of public malware repositories, and the ransomware samples submitted to Anubis [7] since 2011.

The results of our analysis confirm the folk wisdom that such attacks have a continuous increase in the number of families and distinct samples per year [25,37] and also the advances in certain aspects of the specific functionalities of few ransomware families. However, our results also reveal that in a significant number of samples, the core parts of ransomware samples lack the technical complexity to perform successful attacks. While a small fraction of the samples can really prevent the victims from accessing the resources and cause severe problems, a significant number of samples fail to seriously take the victims' resources as hostage. More specifically, we show that more than 94% of ransomware samples in our data set simply try to lock the victims' computer desktop and request ransom, or use very similar and superficial approaches to target the victims' resources.

We also performed an analysis of the charging methods adopted by different ransomware families and also traced the transactions of 1,872 Bitcoin addresses that were used during the `Cryptolocker` attack. The analysis of the transactions shows that cybercriminals started to adopt evasive techniques (e.g., using new addresses for each infection to keep the balances low) in order to better conceal the criminal activity of the Bitcoin accounts. Our analysis also confirms that the Bitcoin addresses used for malicious intents share similar transaction records (e.g., short activity period, small Bitcoin amounts, small number of transactions). However, determining malicious addresses in the Bitcoin network based on the transaction history is significantly difficult, in particular when cybercriminals use multiple independent addresses with small amount of Bitcoins.

In addition to our long-term study, we also evaluate the feasibility of implementing defense mechanisms against destructive ransomware attacks. We provide an analysis of the file system activity of ransomware samples that target users' files. Our analysis shows that different classes of ransomware attacks with multiple levels of sophistication share very similar characteristics from a file system perspective, due to the nature of these attacks. Our analysis suggests that when an infected system is under attack, one can notice a significant change in the file system activity since the malicious process generates a large number of similar file system access requests. Consequently, if we effectively monitor the file system activity (e.g., the changes in Master File Table

(MFT) and the types of I/O Request Packets (IRP) to the file system), it is possible to detect multiple different types of destructive ransomware attacks that target users' files. This contradicts recent discussions in the security community about the impossibility of detecting or stopping these types of attacks due to the use of sophisticated destructive techniques [5,25,34,37]. Based on our analysis, we conclude that detecting and stopping a large number of destructive ransomware attacks is not as complex as it has been reported and deploying practical defense mechanisms against these attacks is possible due to the engineering of NTFS file system.

In summary, the contributions of our paper are as follows:

- We analyzed 1,359 ransomware samples, describing previously undocumented aspects of ransomware attacks with the focus on distinctive and common behaviors among different families.
- We explain how the core parts of ransomware samples are engineered and how these findings can potentially be used to detect these attacks. Our analysis shows that the abnormal file system activity can be accurately monitored in destructive ransomware attacks with different levels of sophistication.
- We perform an analysis of charging methods adopted among ransomware families and also investigate how cybercriminals used cryptocurrency in recent ransomware attacks. Our analysis of illicitly-gained Bitcoins suggests that cybercriminals adopted multiple evasive techniques to protect their privacy in Bitcoin network, making the tracing procedure significantly more difficult.
- We suggest avenues that can be used to defend against a large number of destructive zero-day ransomware attacks. We propose a general methodology to detect these attacks without making any assumptions on how they attack the users' files.

The rest of the paper is structured as follows. In Section 2, we present our data set and ransomware families we categorized. In Section 3, we present experiments we conducted and discuss our findings. In Section 4, we discuss the financial incentives and payment methods. In Section 5, we briefly present related work. Finally, we conclude the paper in Section 6.

2 Ransomware Data Set

Since collecting the malware data set was a critical part of our research, in this section, we provide some details about our ransomware sample selection procedure. To achieve a comprehensive ransomware data set, we collected malware samples from multiple sources. While we obtained 37.9% of our samples from Anubis, 48.3% were collected by automatically crawling public malware repositories [4,2,1]. We captured the remainder 13.8% by manually browsing through security forums [23,3].

After removing the samples that did not execute properly in our environment and those for which we were not able to find a release date, our data set contained a total of 1,359 samples. To obtain accurate labels for these samples, we cross-checked the malware samples by automatically submitting the list of MD5 hashes to VirusTotal. To be conservative on our ransomware malware selection, we consider a malware to be ransomware if at least three AV engines recognized it as belonging to this category.

Table 1: The list of malware families used in our experiments. Some families such as Reveton, Winlock, and Urausy aggressively employed polymorphic techniques.

Family	Family Description				Types of Attacks			
	Samples	Variants	First Seen	Most Recent	Encypting Files	Changing MBR	Deleting Files	Stealing Info
Reveton	244(17.95%)	14	2012	2014			✓	✓
Cryptolocker	32 (2.35%)	4	2013	2014	✓			✓
CryptoWall	11(0.8)	2	2014	2014	✓			
Tobfy	122 (8.97%)	12	2010	2014			✓	
Seftad	23 (1.69%)	4	2006	2010		✓		
Winlock	308(22.66%)	27	2008	2013			✓	
Loktrom	4 (0.29%)	2	2012	2013				
Calelk	9 (0.663%)	2	2009	2010				
Urausy	523 (38.48%)	16	2009	2014			✓	✓
Krotten	17 (1.25%)	3	2008	2009				
BlueScreen	4 (0.29%)	1	2008	2009				
Kovter	8 (0.58%)	2	2013	2013				✓
Filecoder	9 (0.66%)	3	2012	2014	✓		✓	
GPcode	21 (1.54%)	4	2004	2008	✓			
Weelsof	24 (1.76%)	3	2012	2013				
No. of Samples	1,359	-	-	-	73(5.37%)	23(1.69%)	484(35.61%)	44(3.23%)
No. of Variants	-	99	-	-	13(13.13%)	4(4.04%)	29(21.33%)	6(6.06%)

To obtain the family names, we parsed the naming schemes of the AV vendors that are commonly used to assign malware labels. In 77% of samples, AV engines followed the same labeling scheme and our naming policy was mainly based on the popularity of the family name in the community (e.g., `Gpcode`, `Reveton`). The remaining 23% of the samples were labeled in an inconsistent way among the different antivirus software, and in this case we simply selected the most common label among the list of the top 39 AV engines. For example, some samples were labeled both as `Pornoasset` and as `Tobfy` by top AV engines, but we labeled these samples as `Tobfy` due to the perceived popularity of the label.

To the best of our knowledge, our analysis covers the majority of the existing ransomware families observed between 2006 to 2014. However, as our data collection module relies on external sources, we are aware of the possibility of missing some types of ransomware attacks. Furthermore, in order to conduct balanced experiments over the ransomware families, and also to avoid biased results due to polymorphic techniques, we performed our analysis not only based on individual samples, but also based on the families and distinct variants per family. Table 1 shows the total number of distinct samples per family as well as distinct variants in each family. It also shows the first time they appeared in the wild and the most recent sample in our data set.

As it can be clearly seen from Table 1, there is a rapid emergence of new families between 2012 and 2014, as well as a significant growth on the number of new samples in each family. This may be due to a bias on the data set toward more recent samples, or to the multiplication of samples due to polymorphism in newer families. (e.g., `Winlock`, `Urausy`, and `Reveton`). The Table also shows the types of ransomware attacks we observed among each family in our data set (in addition to locking the user desktop). In particular, we observed that 61.22% of the samples (57 variants) only targeted the desktop of compromised computers, without touching the documents in the file system. More details on the locking procedure are discussed in Section 3.1. Encrypting the victim files in addition to locking the desktop was observed in 5.37% of samples in four families (`Cryptolocker`, `CryptoWall`, `Filecoder`, and `Gpcode`). We also observed the emergence of other malicious activities, such as changing the browser setting or performing multiple infections to install other malware, in 3.23% of the sam-

ples. Despite the fact that the number of samples performing additional malicious activities (e.g., stealing private information) is not alarmingly high, this phenomenon is now increasing. For example, our analysis shows that information stealing was first seen in `Reveton` in early 2012, but other families such as `Kevtor`, `Urausy`, and `Cryptolocker` started to add stealing information capabilities to their samples after that date [16,21]. We provide more details on the malicious behaviors among ransomware families in Section 3.

2.1 Experimental Setup

We performed all malware execution experiments according to common scientific guidelines [33] inside a Cuckoo Sandbox [14] running Windows XP SP3 32bit, with a controlled access to the Internet via NAT. Network traffic (e.g., IRC, DNS and HTTP) were allowed to enable commands and controls (C&C) communication. In order to control harmful traffic (e.g., spam) during the execution of the experiments, we redirected this traffic to a local honeypot. The network bandwidth was also reduced to mitigate potential DoS attacks.

We then executed each sample in the analysis environment for 45 minutes to capture the execution traces of the sample. Since current ransomware samples typically start attacking the user's files right after the malicious program is executed by the user, we believe that the 45 minutes threshold is sufficient for most ransomware samples to exhibit their malicious behavior. After each execution, the entire system is rolled back to a clean state to prevent any interference across executions.

3 Characterization and Evolution

In this section, we describe our findings based on the types of malicious activities detected in ransomware samples during our experiments. We partition the malicious activities into multiple categories and discuss our findings in each of them.

3.1 File System Activity

One of our first goals was to describe how a malicious process interacts with the file system when a compromised computer is under a ransomware attack. To answer this question, we investigate the common characteristics of ransomware attacks from a file system perspective regardless of the technical differences that these attacks might have (such as the infection and the key generation techniques). In order to monitor the file system activity, multiple approaches could be used. One classic approach is to hook the SSDT table [19,22] to monitor interesting function calls. In our analysis, we developed a minifilter driver [26] to capture all I/O Request Packets (IRP functions for all access modes) that the I/O manager generates on behalf of user-mode processes to access the file system.

To monitor the I/O requests the minifilter driver registers callback routines to the filter manager. In our analysis, we defined pre-operation and post-operation callback routines for all IRP functions in order to precisely record any I/O and transaction activity on the

files. For each file system request, we collected the process name, the process ID, the parent process ID, the pre-operation and post-operation callback time, the IRP type, the arguments and the result of the operation. Each record is a tuple:

<PName,PID,PPID,PreOpTime,PostOpTime,IRPFlag,Args,Result>

The minifilter with different callback routines allows us to capture all the the read, write, and attribute change requests to the file system at the closest possible level to the file system driver. Our minifilter driver is deployed in a privileged kernel mode that has access to nearly all objects of the operating system. Furthermore, since we captured the file system activity directly from the I/O manager in the kernel, there was a low chance that cybercriminals could bypass our monitor. When looking at the execution traces of the malware program in the analysis environment, we observed that the way malicious processes generate requests to access file system was significantly different from benign processes. By performing a close examination of the file system activity of multiple ransomware samples, we were able to distinguish multiple attack strategies that ransomware families used while the system was under the attack. We discuss our findings in the following sections.

Encryption Mechanisms As presented in Table 1, 5.37% of the samples among four families employed some encryption mechanisms during the experiments. Our analysis shows that existing ransomware samples use both customized and standard cryptosystems during the attacks. The customized cryptosystems are not necessarily more reliable or complicated than the standard cryptosystems that Windows platforms provide (e.g., `CryptoAPI`). Cybercriminals develop their own cryptosystems for multiple reasons. One reason is probably to decrease the chance of being easily detected by common malware analysis techniques (e.g., PE header checking, Hooking standard API functions). One of the key features crypto-style ransomware samples should have is to reliably minimize the chance of recovering the original data after generating the encrypted files. Some of the modern crypto-style ransomware families such as `cryptolocker` and `CryptoWall` make use of standard Windows functions to perform their file encryption. They simply call `CryptEncrypt` with an handle to the encryption key and a pointer to a buffer that contains the plaintext to be encrypted. In these families, the plaintext in the buffer is directly overwritten with the encrypted data created by this function. As depicted in Table 2, the I/O manager generates `IRP_MJ_CREATE` on behalf of the malicious process to open the user's file. The file content is read via `IRP_MJ_READ` for encryption and is overwritten with the ciphertext buffer using the `IRP_MJ_WRITE` function each time a file encryption occurs.

We also observed that even if the samples do not use standard cryptosystems, it is still possible to recognize how they attack users' files. For instance, a member of the `Filecoder` family uses a simple customized approach to encrypt files. Unlike `Cryptolocker` and `CryptoWall`, the sample first generates an encrypted version of a file using an AES-256 encryption key and then overwrites the original file's data with the encrypted file. Table 3 shows how the malicious process interacts with the file system to encrypt an arbitrary file when the system is under the attack. The types of IRPs generated when the malicious process operates show how a ransomware sample targets the victim's files. For example, the sequence of IRPs shows that the ransomware

Table 2: The IRP requests generated on behalf of the malicious process during Cryptowall attack. Similar file system activity traces were also observed in Cryptolocker attack due to the use of Windows standard cryptosystem. The table does not show all the captured information for each I/O operation.

Process Name	Operation	Path	Result
mal.exe	IRP_MJ_CREATE	E:\MySubmissions	SUCCESS
mal.exe	IRP_MJ_DIRECTORY_CONTROL	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_READ	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_WRITE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_READ	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_WRITE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
.			
.			
.			
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_SET_INFORMATION	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS

Table 3: The types of IRPs requested by a malicious process to encrypt and overwrite the victim's files during a ransomware attack. The attack strategy can be detected by analyzing the I/O requests sent to the file system.

Process Name	Operation	Path	Result
mal.exe	IRP_MJ_CREATE	E:\MySubmissions	SUCCESS
mal.exe	IRP_MJ_DIRECTORY_CONTROL	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS
mal.exe	IRP_MJ_READ	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_READ	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
.			
.			
mal.exe	IRP_MJ_WRITE	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS
.			
.			
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
.			
.			
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_SET_INFORMATION	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
.			
.			
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS
mal.exe	IRP_MJ_SET_INFORMATION	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\dimva2015-submission.tex.crypt	SUCCESS

sample first queries the given location to find the user's file and creates handles to the original and encrypted files. The file's data is read via a IRP_MJ_READ IRP and the encrypted data buffer is written to the destination file via a IRP_MJ_WRITE IRP. Consequently, IRP_MJ_SET_INFORMATION is used to delete the original file after the file is closed and also to overwrite the original file with the encrypted file. The sequence of IRPs shown in Figure 3 is repeated for every file on the infected system.

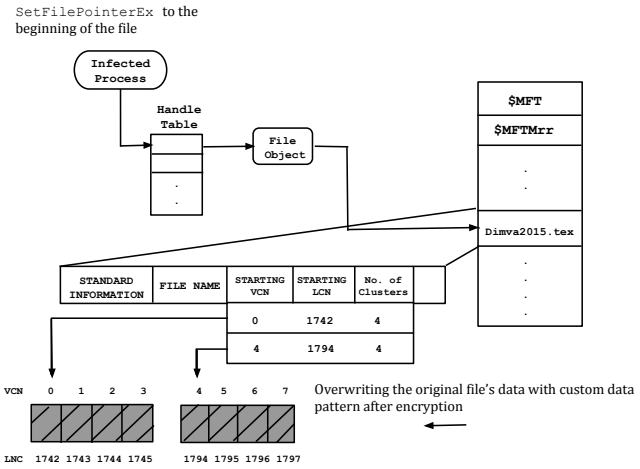


Fig. 1: The malicious process attempts to get the file map on the physical disk in order to overwrite the file's data after the encryption.

Another sample from `Filecoder` makes use of the `Defragmentation API` to get raw access to each file's data based on the volume sector and the cluster size. The sample overwrites the files with custom data patterns based on how the files are kept on the disk. For example, if the file mapping check shows that the file has multiple extents, the physical disk offsets of each extent should be retrieved to be overwritten with the custom data pattern. If the file does not have any extents, it means that the file is small and is kept as a MFT entry in the MFT table. The malware uses the `DeviceIoControl` from `kernel32.dll` to get the file map on the physical disk. Figure 1 shows how a malicious process finds the file's data and overwrites the data after the encryption. When NTFS finds the file record for the MFT, it obtains the VCN-to-LCN mapping information in the file records data attribute. Consequently, the malicious process can easily retrieve the information and locate the file's data on the disk.

Encryption techniques (e.g., key generation and key management) in crypto-style ransomware families have also evolved significantly. For example, a `Gpcode` variant generates a static key during the attack. This key is also used to encrypt all the non-system files. Finding the encryption key in this variant is fairly simple and we were able to retrieve the key by comparing the encrypted file and the original one. The most recent `Gpcode` variant in our data set encrypts the files using a unique AES-256 encryption key. The encryption key is then encrypted using a 1024-bit RSA public key. Another change we observed over time is the place where an asymmetric key pair is generated. For example, in a sample (`md5:ffcf2bb69f23c7c234d2f2ee380cdaa4`) created in 2012, the master key is generated locally in the compromised computer and can be extracted by looking into the memory. The use of RSA keys with different key length in `Cryptolocker` was previously reported [30], but at the time of writing, we observed only samples with 1024-bit RSA public key in our data sets. The RSA public key is generated remotely on the C&C server once the compromised

Table 4: A set of IRP requests generated on behalf of a malicious process to delete files during an attack. The process simply searches the directory, creates a handle to files and deletes them via IRP_MJ_SET_INFORMATION.

Process Name	Operation	Path	Result
mal.exe	IRP_MJ_DIRECTORY_CONTROL	E:*	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\	SUCCESS
mal.exe	IRP_MJ_CREATE	E:\	SUCCESS
mal.exe	IRP_MJ_DIRECTORY_CONTROL	E:\MySubmissions*	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\	SUCCESS
mal.exe	IRP_MJ_CREATE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_DIRECTORY_CONTROL	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_SET_INFORMATION	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLEANUP	E:\MySubmissions\dimva2015-submission.tex	SUCCESS
mal.exe	IRP_MJ_CLOSE	E:\MySubmissions\dimva2015-submission.tex	SUCCESS

computer successfully sends a POST request to C&C servers. If the sample cannot connect to C&C servers, the malicious behavior is not triggered. The sample md5: 04fb36199787f2e3e2135611a38321eb only encrypted users' files in logical drives introduced in the system. An evolution in this family is the encryption of connected drives. The sample (md5:f1e2de2a9135138ef5b15093612dd813) encrypts all non-system files including network shares to minimize the possibility of recovering files without paying the ransom. These ransomware samples simply employ GetLogicalDrives, GetDriveType or similar functions to find network drives.

Deletion Mechanisms In this part, we specifically discuss file deletion mechanisms that are unique to ransomware attacks. 35.6% of samples among five common ransomware families do not perform any encryption mechanisms. Instead, they delete the user's files if the user does not pay the ransom. On the other hand, we observed that certain samples in Gpcode and Filecoder families deleted the original unencrypted file's data after the encryption occurred. Consequently, deletion operation is a common task among multiple ransomware families in our data set. Table 4 shows a sequence of IRPs collected while running a sample from the Filecoder family. The malicious process uses the IRP_MJ_DIRECTORY_CONTROL function to list the files and then requests to open the file via a Win32 CreateFile. Any create requests are performed by IRP_MJ_CREATE function which returns a handle to the file objects. Finally, the file is deleted by IRP_MJ_SET_INFORMATION when the file is closed. We observed very similar approaches in other families such as Gpcode, Reveton and Urausy in spite of differences in other aspects of the attacks.

In the NTFS file system, each file has an entry in the Master File Table (MFT) that reflects the changes of the corresponding file or folder [10]. The core file's attributes in each MFT entry can be found in the $STANDARD_INFORMATION attribute, and the $DATA attribute that contains the content of the corresponding file. The content of the $DATA attribute could be resident or non-resident in the MFT entry depending on the size of a file. Figure 2 shows the disk layout for files with different sizes in the NTFS file system. The status of a file is determined by both a flag and a $BITMAP in an MFT entry. $BITMAP manages the information about allocation status of clusters within the disk.

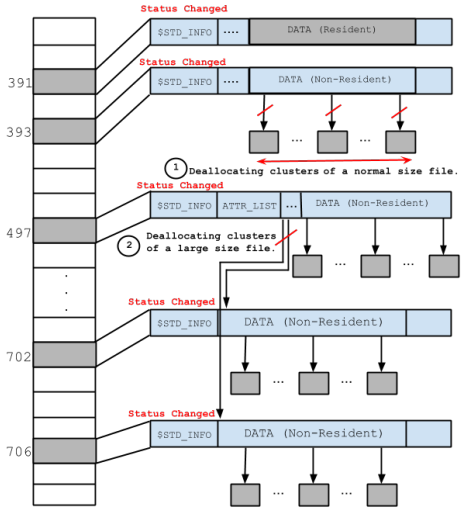


Fig. 2: Disk layout for files with different sizes in NTFS file system. The content of large files is defined as Non-resident `$DATA` and is managed by a runlist attribute in each MFT entry. During a ransomware attack, the clusters are deallocated and the status of the file is changed.

When a ransomware attack occurs, the malware lists the non-system files and initiates a delete operation for each of them. The MFT entry for each file is updated by changing the status flag value of the file from `0x01` to `0x00`. Furthermore, the `$BITMAP` attribute in MFT file is set to zero for the corresponding file. For large files, since multiple clusters might be allocated, the location of fragmented data is saved in the `runlist` in the header of MFT entry. When the file is deleted, the clusters that are used to keep the file's data are set to unallocated in `$BITMAP` attribute in the MFT file. Consequently, when a file is deleted in a typical ransomware attack, the MFT entry is updated, but the content of the file is not deleted immediately. Therefore, our analysis suggests that we can detect ransomware attacks that target users' files based on the changes in the MFT table and also recover the content associated with the deleted files due to the engineering of the NTFS file system. Finally, Figure 3 shows the delete operation from a different perspective- when the malicious process tries to delete a large file that is fragmented among multiple clusters.

Changing Master Boot Records One of the ransomware families (`Seftad`) was developed to attack the Master Boot Records (MBR) which contains the executable boot code and the partition table. The MBR is located on the first sector of a hard disk, and it is loaded into memory at boot time when the system transfer control to the code stored in the MBR. Samples that target the MBR prevent the infected system from loading the boot code in the active partition by simply replacing it with a bogus MBR that displays a message asking for a ransom. Defeating this type of ransomware attack is quite simple. For example, in early samples, the unlock code was hard-coded into the binary and could be acquired by reverse engineering. Following this procedure, we discovered the unlock code in 18 `Seftad` samples in our data set.

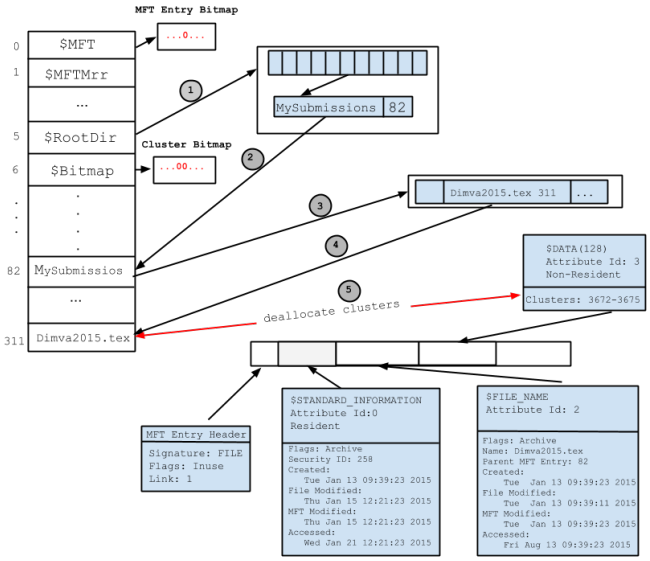


Fig. 3: A ransomware attack(`Gpcode`) with a simple delete operation. The clusters used to keep the file `$data` are deallocated in an MFT entry.

Locking Procedure An important step in a successful ransomware attack is to lock the desktop of the computer under attack. This is typically done by creating a new desktop and making it persistent. Ransomware samples simply use `CreateDesktop` to create a fresh desktop environment and eliminate unnecessary processes. The new desktop is created via a `DESKTOP_SWITCHDESKTOP` access mode that enables the `SwitchDesktop` function to activate the new desktop and receive input from the victim. The desktop is assigned to a thread using the `SetThreadDesktop` function. A significant number of samples in our data set (61.22%) use very similar approaches to establish a persistent desktop lock.

A small number of samples (8 variants) in families like `Urausy`, `Reveton`, and `Winlock` employed another approach to lock the desktop. In these families, the lock banner is simply downloaded as a HTML page with corresponding images based on the victim's geographical location and it is then displayed in full screen in a `IE` window with hidden controls. The banner plays a local law enforcement warning in the language used in the victim's geographical location. The warning typically says that the operating system is locked due to infringement against certain laws (e.g., distributing copyrighted materials or visiting child pornography sites) in that location.

Disabling certain keyboard shortcuts such as toggling (e.g., Windows key + `Tab`) is automatically done once a new desktop is created because no other applications are open to toggle through. However, disabling special keys is another part of the locking procedure. This is done by installing hook procedures that monitor keyboard input events. The number of disabled keys was different in different ransomware families. For example, 18 variants in `Reveton` and `Urausy` disabled Windows keys to prevent the victims from entering the start menu and 72 variants among 15 families attempted to

disable the Esc Key to prevent the victims from using keyboard shortcuts (e.g., starting Windows Task Manager) during the attack.

3.2 Mitigation Strategies

API Call Monitoring As discussed in Section 3.1, a significant number of ransomware samples use Windows API functions to lock the victim's desktop. Those API calls can be used to model the application behavior and train a classifier to detect suspicious sequence of Windows API calls. This approach is not necessarily novel, but it would allow us to stop a large number of ransomware attacks that are produced with little technical efforts. For example, a sequence of `GetThreadDesktop`, `CreateDesktopW` and `SwitchDesktop` functions can be converted to a sequence of API calls. Of course, cybercriminals might be able to evade detection using different techniques. For example, they may use native APIs to directly lock the system under the attack. However, the implementation of such ransomware samples requires significant work since the native APIs are not properly documented and may change among different versions, which can limit the portability of the attack.

Monitoring File System Activity Our analysis also suggests that it is possible to detect ransomware attacks – even the ones using deletion and encryption capabilities – based on our findings in Section 3.1. Our analysis shows that significant changes occur in the file system activities (e.g., a large number of similar encryption, deletion requests) when the system is under a ransomware attacks. By closely monitor the MFT table, one can detect the creation, encryption or deletion of files. For example, when the system is under a ransomware attack, a significant number of status changes occur in a very short period of time in MFT entries of the deleted files. For encrypted files, we notice a large number of MFT entries with encrypted content in the `$DATA` attribute of files that do not share the same path (e.g., files within a directory). In our definition, a malicious MFT entry is a MFT entry that is generated or modified in a system under a ransomware attack. A classifier can be trained on benign and malicious MFT entries to detect abnormal file system activities when the system is under an attack.

In order to distinguish between benign and malicious file system activity, another possible approach consists of monitoring all I/O requests that user-mode processes generate. A system with protection capabilities can intercept all the file system requests and discard the suspicious requests before they reach the file system driver.

Recovering the deleted files from the ransomware attacks would also be possible. If the `$DATA` attribute is resident in the MFT entry, the content of the file can be simply copied to another location. For non-resident `$DATA` attributes, we need to parse the `RunList` in the MFT entry and copy the raw data to another location and perform the recovery. In any case, early detection of the attack is critical in order to successfully recover the content of deleted files, since the deallocated clusters can be allocated to new files and the content of the deleted file will be overwritten. This approach can be applied to most of the ransomware samples with either customized or standard cryptosystems since the file level activity is a common characteristic of ransomware samples that target users' files.

Using Decoy Files The attack strategies adopted to encrypt or delete the user files are very similar among ransomware families. For example, the malicious process aggressively attacks all files (in different paths, and with different extensions) and tries to encrypt and/or delete them in a very short period of time. Therefore, defining a file system activity model that reflects the normal interaction with the file system is possible. However, cybercriminals could try to evade detection by launching attacks while mimicking a normal user behavior. For example, a cybercriminal may avoid aggressively encrypting all files and starts by encrypting files with recent access or modification time. Approaches like this might not be detected by approaches that monitor the behavior of the system. However, one technique to detect these attacks could be to install decoy files in multiple locations of the disk that are constantly monitored. The use of decoy resources to detect security breaches and insider attacks was first proposed in [9,40]. Decoy resources have also been recently used to improve the security of hashed passwords [20] and to detect illegally obtained data from file hosting services [28]. In our definition, monitoring decoy files can be an additional layer of defense on top of the I/O requests monitoring to detect ransomware attacks.

It is possible to generate decoy files in a way that it is computationally difficult for an adversary to discern them based on the files' attributes. The decoy files should be indexed at multiple places of any possible directory listing that a malicious process can programmatically do (e.g., listing based on name, create/modification/access time, size, content, etc). This approach can increase the chance of decoy files being touched by the malicious process in early stages of the attacks regardless of the fact that the ransomware sample uses novel strategies or customized/standard cryptosystems.

4 Financial Incentives

Since the ultimate goal of ransomware attacks is to get money from victims, the payment method is an important aspect of the attacks. Cybercriminals continuously strive to find more reliable charging methods by improving two important properties: (1) the difficulty of tracing the recipient of the payments, and (2) the ease of exchanging payments into a preferred currency. Table 5 provides a breakdown of the charging methods used by ransomware families over the past years. Our analysis suggests that sending SMS to premium numbers is not necessarily used in old types of ransomware attacks. For example, the charging method in `Calelk` is still based on using premium numbers. The premium rate numbers were hard-coded in the ransomware sample or were downloaded from the C&C servers in each infection. This class of ransomware attacks requires the least amount of technical background and when propagated in a large scale the revenue could be significant.

A large fraction of ransomware samples (88.22%) used prepaid online payment systems such as Moneypak, Paysafecard, and Ukash cards, since they provide limited possibilities to trace the money. These services are not tied to any banking authority and the owner of the money is anonymous. The ransomware business model takes advantage of these systems since there are no records of the vouchers to trace cybercriminals. In a typical scenario, once a ransomware criminal receives the vouchers, in order to monetize them, he can sell vouchers in underground voucher exchange forums, ICQ, or

Table 5: Summary of types of charges in 15 ransomware families.

Families	Type of Charge			
	Premium Number	Untraceable Payments	Online Shopping	Bitcoin Transactions
Reveton		✓	✓	
Cryptolocker		✓		✓
CryptoWall				✓
Tobfy		✓		
Seftad	✓			
Winlock				
Loktrom	✓			
Calelk	✓			
Urausy		✓	✓	
Krotten		✓		
BlueScreen		✓		
kovter		✓	✓	
Filecoder		✓		
GPcode		✓		
Weelsof		✓		
Number of Samples	132 (9.71%)	1,199 (88.22%)	14(1.03%)	28 (2.86%)
Number of Variants	18 (19.35%)	75 (80.64%)	4 (4.30%)	4 (4.3%)

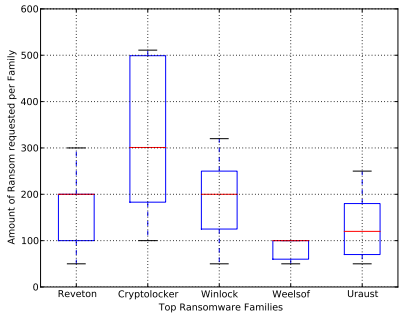
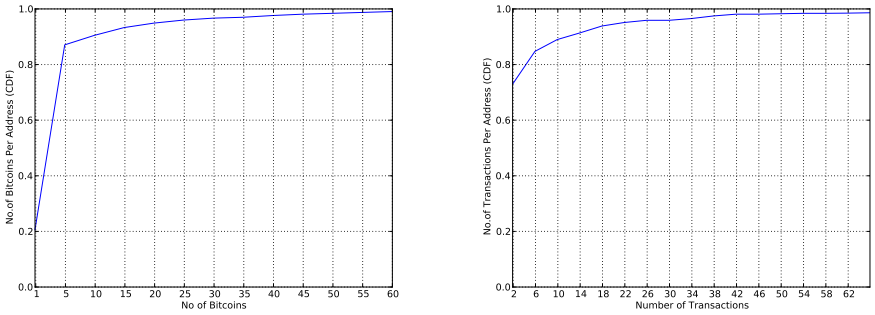


Fig. 4: The amount of ransom money among common ransomware families. Around 89.2% of Cryptolocker victims paid more than 100 dollars. One reason is the significant increase in the value of Bitcoin between mid-September and mid-November. We observed more changes in the amount of requested ransom in Cryptolocker probably due to low stability in Bitcoin exchange rates.

other untraceable communication channels for a lower price than the nominal value of the vouchers. We also found some unconventional methods used for charging victims. We found two variants of `Kevtor` family that forced users to buy a software package which unlocked the compromised computer. Figure 4 represents the amount charged per family based on our data set. The amount of money required by ransomware owners to unlock the computer changes based on variants and families. For examples, 48.43% of samples among top six families demanded between 150 to 250 dollars.

4.1 Bitcoin as a Charging Method

Bitcoin provides some unique technical and privacy advantages for miscreants behind ransomware attacks. Bitcoin transactions are cryptographically signed messages that embody a fund transfer from one public key to another and only the corresponding



(a) The number of Bitcoins per address.

(b) The total number of transactions per Bitcoin address

Fig. 5: 20.1% of Bitcoin addresses received no more than one Bitcoin probably because victims were charged less due to a dramatic increase of Bitcoin value in late November 2013. Furthermore, approximately 73% of Bitcoin addresses had only two transactions. The incoming transaction is made by victims to pay the ransom and the outgoing transaction is performed by the ransomware owner to send the Bitcoin to another addresses in order to make tracing infeasible.

private key can be used to authorize the fund transfer. Furthermore, Bitcoin keys are not explicitly tied to real users, although all transactions are public. Consequently, ransomware owners can protect their anonymity and avoid revealing any information that might be used for tracing them.

We performed an analysis of the use of Bitcoins in recent ransomware attacks where victims had to buy Bitcoins in order to access their resources. We acquired the Bitcoin addresses by searching the web as well as public forums [31] that conducted discussions on `Cryptolocker` attacks. Victims typically participated in the discussions by posting information about their infection and the Bitcoin addresses to which they were required to send the ransom. We collected 1,872 Bitcoin addresses during the experiments. We automatically queried the transactions from publicly accessible Bitcoin block explorer websites [8] and parsed the results into a database.

The number of Bitcoins collected by cybercriminals during `Cryptolocker` attack is previously reported [35]. Our main focus in this part is to provide insights into how cybercriminals employed Bitcoin to collect the ransom fee based on the transactions history. One of the questions we wanted to answer was whether it is possible to detect illicitly-gained Bitcoins based on the transaction history of a Bitcoin address. Our analysis suggests that identifying these Bitcoins is getting significantly difficult since cybercriminals have started to use evasive approaches to protect their privacy (e.g., multiple independent Bitcoin addresses, small Bitcoin amounts, short activity period, small transaction records) after receiving large volumes of Bitcoins from victims. One reason to use multiple independent addresses with small Bitcoin amounts could be that concealing the source of thousands of illicitly-obtained Bitcoins is a critical task if cybercriminals want to transfer the Bitcoins via recognized exchanges without being noticed. In fact, this is the main evolution in employing Bitcoin in ransomware attacks to make the potential tracing procedures more difficult in the Bitcoin network.

Our analysis on Bitcoin transactions shows that 84.46% of Bitcoin addresses had no more than six transactions. Furthermore, a significant fraction of these Bitcoin addresses (68.93%) were active for at most 10 days. These addresses were directly used to receive Bitcoins from victims. Another type of addresses had more transactions and were active for a longer period of time (e.g., more than 10 days). These addresses were used to aggregate the collected ransom fees. Figure 5(a) shows the CDF of number of Bitcoin per Bitcoin address. In 48.9% of Bitcoin addresses that we analyzed, a Bitcoin address received at most two Bitcoins. These transactions have occurred in early steps of the attacks when two Bitcoins were worth roughly 200 dollars equal to the ransom fee required by cybercriminals to send the decryption key.

As shown in Figure 5(b), approximately 72.9% of Bitcoin transactions belong to Bitcoin addresses with two transactions. The incoming transaction was made by victims to pay the ransom and, the outgoing transaction was performed by cybercriminals. The collected Bitcoins were transferred through tens of temporary intermediate accounts or split into many small amounts in order to be recombined in a new account later to decrease possibilities of tracing the money.

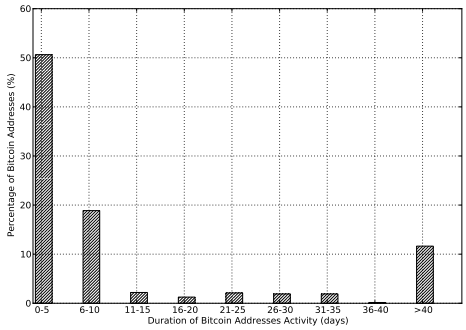


Fig. 6: The duration of activity for Bitcoin addresses. Approximately 50% of Bitcoin addresses have zero to five days of active life.

As provided in Figure 6, our observation also suggests that Bitcoin addresses that were used to collect Bitcoins from victims have a relatively short duration of activity. This is due to the fact that the accumulated Bitcoins had to be transferred to other accounts within a few hours or a few days probably to use mix services and conceal the source of the money.

5 Related Work

Ransomware and Underground Economy Various security vendors have reported the threat potential of ransomware attacks based on the number of infections that they observed [6,29,37]. The use of cryptography to mount extortion based attacks was first introduced in [38]. Employing Microsoft Cryptographic API (MS CAPI) calls to design cryptovirus samples was presented by Young [39]. Young demonstrated how to use MS CAPI to generate keys and encrypt the user's data.

The first step to analyze specific ransomware families was made by Gazet by analyzing three primitive ransomware families [18]. He concluded that while these early families were designed for massive propagation, they did not fulfill the basic requirements (e.g., sufficiently long encryption keys) for mass extortion.

The presence of scareware as rogue security software has been also studied over the past few years. Stone-Gross et al. performed an analysis of underground economy of fake antivirus software. They built an economic model that showed how cybercriminals performed refunds and chargebacks in order to conceal their criminal nature for a longer period of time [36]. Cova et al. provided an analysis of fake antivirus structure and measured the number of victims and the profits gained based on the web servers used by several fake antivirus groups [13].

Bitcoin Privacy Bitcoin has also recently received considerable interest regarding the security and anonymity in security research. Meiklejohn et al. developed a clustering heuristic that was used to cluster Bitcoin addresses belonging to a particular user [24]. They discussed the potential anonymity in the Bitcoin protocol and the actual anonymity achieved by users. Reid et al. constructed two graphs based on publicly available transaction history [17]. They used the properties of these graphs to illustrate how information leakage can be used to de-anonymize the system's users. Using this technique, they described the flow of stolen money from MyBitcoin. Recently, Ron et al. performed an analysis over the user graph and provided an in-depth analysis of the largest transactions in Bitcoin history [32]. In another work, Möser performed an analysis of the anonymity and transaction graph of three Bitcoin mix services. He found that all the three Bitcoin mix services had a distinct transaction graph pattern, but some of them were more successful than others [27]. In order to characterize the popularity of illicit goods, Christin performed an analysis by extracting data from Silk Road marketplace [11]. Although the work does not examine the Bitcoin block chain, it provides an estimation of the market value of such transactions.

A closer and concurrent work to our interest was performed by Spagnuolo et al. that parsed the blockchain and clustered the Bitcoin addresses that were likely to belong to certain users or groups [35]. They labeled the users based on the information that was scraped from openly available resources. They were able to label Bitcoin addresses on real-world cases such as Silk Road and Cryptolocker ransomware. We also used public repositories to extract Bitcoin addresses that belong to cybercriminals behind ransomware attacks. However, unlike Stagnuolo et al. work [35], our goal is to characterize the Bitcoin addresses used for malicious intents based on the transaction history rather than de-anonymizing the Bitcoin users.

6 Conclusion

In this paper, we performed a long-term analysis of ransomware families with a special focus on their destructive functionality. The characterization of ransomware attacks was based on 1,359 ransomware samples among 15 families that have emerged over the last few years. Our results show that a significant number of ransomware families share very similar characteristics in the core part of the attacks, but still lack reliable destructive functions to successfully target victims' files.

We also describe how a malicious process interacts with the file system when a compromised computer is under a ransomware attack. We observed that suspicious file system activity of multiple types of destructive ransomware families can be reliably monitored. When looking at the execution traces of the malware programs, we observed that the way malicious processes generate requests to access file system was significantly different from benign processes. We also observed that different classes of ransomware attacks with multiple levels of sophistication share very similar characteristics from file system perspective due to the nature of these attacks. Unlike recent discussions in security community about ransomware attacks, our analysis suggests that implementing practical defense mechanisms is still possible, if we effectively monitor the file system activity for example the changes in Master File Table (MFT) or the types of I/O Request Packets (IRP) generated on behalf of processes to access the file system. We propose a general methodology that allow us to detect a significant number of ransomware attacks without making any assumptions on how samples attack users' files.

References

1. Minotaur Analysis - Malware Repository. minotauranalysis.com.
2. VX Vault - Online Repository of Malware Samples. vxvault.siri-urz.net.
3. Malware Tips - Your Security Advisor. http://malwaretips.com/forums/virus-exchange.104/.
4. MalwareBlackList - Online Repository of Malicious URLs. http://www.malwareblacklist.com.
5. Police ransomware threat assessment. Europol Public Information, 2014.
6. AJJAN, A. Ransomware: Next-Generation Fake Antivirus. http://www.sophos.com/en-us/medialibrary/PDFs/technicalpapers/SophosRansomwareFakeAntivirus.pdf, 2013.
7. BAYER, U., KRUEGEL, C., AND KIRDA, E. TTAnalyze: A Tool for Analyzing Malware. In *Proceedings of the European Institute for Computer Antivirus Research Annual Conference* (April 2006).
8. BLOCKCHAIN.INFO. Bitcoin Block Explorer. https://blockchain.info.
9. BOWEN, B. M., HERSHKOP, S., KEROMYTIS, A. D., AND STOLFO, S. J. *Baiting inside attackers using decoy documents*. Springer, 2009.
10. CARRIER, B. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.
11. CHRISTIN, N. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of WWW 2013* (May 2013).
12. CISCO, INC. Ransomware on Steroids: Cryptowall 2.0. http://blogs.cisco.com/security/talos/cryptowall-2, 2015.
13. COVA, M., LEITA, C., THONNARD, O., KEROMYTIS, A. D., AND DACIER, M. An Analysis of Rogue AV Campaigns. In *Proceedings of the International Conference on Recent Advances in Intrusion Detection* (2010), pp. 442–463.
14. CUCKOO FOUNDATION. Cuckoo Sandbox: Automated Malware Analysis. www.cuckoosandbox.org, 2014.
15. DELL SECUREWORKS. Cryptolocker Ransomware. http://www.secureworks.com/cyber-threat-intelligence/threats/cryptolocker-ransomware/, 2014.
16. DONOHUE, B. Reveton Ransomware Adds Password Purloining Function. http://threatpost.com/

reveton-ransomeware-adds-password-purloining-\function/100712, 2013.
17. FERGAL, R., AND MARTIN, H. An analysis of anonymity in the bitcoin system. In *Security and Privacy in Social Networks* (2012).
18. GAZET, A. Comparative analysis of various ransomware virii. *Journal in Computer Virology 6*, 1 (February 2010), 77–90.
19. HOGLUND, G., AND BUTLER, J. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional, 2005.
20. JUELS, A., AND RIVEST, R. L. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 145–160.
21. KREBS, B. Inside a Reveton Ransomware Operation. http://krebsonsecurity.com/2012/08/inside-a-reveton-ransomware-operation/, 2012.
22. LANZI, A., BALZAROTTI, D., KRUEGEL, C., CHRISTODORESCU, M., AND KIRDA, E. Accessminer: Using system-centric models for malware protection. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (2010), CCS '10, ACM, pp. 399–412.
23. MALWARE DON'T NEED COFFEE. Guess who's back again ? Cryptowall 3.0. http://malware.dontneedcoffee.com/2015/01/guess-whos-back-again-cryptowall-30.html, 2015.
24. MEIKLEJOHN, S., POMAROLE, M., JORDAN, G., LEVCHENKO, K., MCCOY, D., VOELKER, G. M., AND SAVAGE, S. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (2013), IMC '13, pp. 127–140.
25. MICROSOFT, INC. Microsoft Security Intelegence Report Vol. 16. http://www.microsoft.com/security/sir/default.aspx, 2013.
26. MICROSOFT, INC. File System Minifilter Drivers. https://msdn.microsoft.com/en-us/library/windows/hardware/ff540402%28v=vs.85%29.aspx, 2014.
27. MÖSER, M. Anonymity of bitcoin transactions: An analysis of mixing services. In *Proceedings of Monster Bitcoin Conference* (2013).
28. NIKIFORAKIS, N., BALDUZZI, M., ACKER, S. V., JOOSEN, W., AND BALZAROTTI, D. Exposing the lack of privacy in file hosting services. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats (LEET)*, LEET 11.
29. O'GORMAN, G., AND MCDONALD, G. Ransomware: A Growing Menance. http://www.symantec.com/connect/blogs/ransomware-growing-menace, 2012.
30. PRINCE, B. CryptoLocker Could Herald Rise of More Sophisticated Ransomware. http://www.darkreading.com/attacks-breaches/cryptolocker-could-herald-rise-of-more-sophis\ticated-ransomware, 2013.
31. QUICKBT. Disturbing Bitcoin Virus. http://www.reddit.com/r/Bitcoin/comments/1o53hl/, October 2013.
32. RON, D., AND SHAMIR, A. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security 2013* (2013), vol. 7859, pp. 6–24.
33. ROSSOW, C., DIETRICH, C. J., GRIER, C., KREIBICH, C., PAXSON, V., POHLMANN, N., BOS, H., AND VAN STEEN, M. Prudent practices for designing malware experiments: Status quo and outlook. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 65–79.
34. SOPHOS, INC. Security Threat Report 2014, Smarter, Shadier, Stealthier Malware. http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-security-threat-report-2014.pdf, 2014.

35. SPAGNUOLO, M., MAGGI, F., AND ZANERO, S. BitIodine: Extracting intelligence from the bitcoin network. In *Financial Cryptography and Data Security* (March 2014), Lecture Notes in Computer Science (LNCS), Springer-Verlag.
36. STONE-GROSS, B., ABMAN, R., KEMMERER, R. A., KRUEGEL, C., STEIGERWALD, D. G., AND VIGNA, G. The Underground Economy of Fake Antivirus Software. In *Proceedings of the Workshop on the Economics of Information Security and Privacy* (2013).
37. SYMANTEC, INC. Internet Security Threat Report. `http://www.symantec.com/security_response/publications/threatreport.jsp`, 2014.
38. YOUNG, A., AND YUNG, M. Cryptovirology: Extortion-based security threats and countermeasures. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on* (1996), IEEE, pp. 129–140.
39. YOUNG, A. L. Building a Cryptovirus Using Microsoft's Cryptographic API. In *Proceedings of the International Conference on Information Security* (2005), pp. 389–401.
40. YUILL, J., ZAPPE, M., DENNING, D., AND FEER, F. Honeyfiles: deceptive files for intrusion detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC* (2004), IEEE, pp. 116–122.