Ransomware behavioural analysis on windows platforms

Nikolai Hampton

Zubair A. Baig
Edith Cowan University, z.baig@ecu.edu.au

Sherali Zeadally

10.1016/j.jisa.2018.02.008

Ransomware Behavioral Analysis on Windows Platforms

Nikolai Hampton[1], Zubair Baig[2], Sherali Zeadally[3]
nikolaih@3583bytesready.net, z.baig@ecu.edu.au, szeadally@uky.edu
[1]Impression Research, Brisbane, Australia
[2]School of Science and Security Research Institute, Edith Cowan University, Joondalup 6027, Australia
[3]College of Communication and Information, 315 Little Library Bldg, University of Kentucky, Lexington, KY 40506-0224, USA

Abstract

Ransomware infections have grown exponentially during the recent past to cause major disruption in operations across a range of industries including the government. Through this research, we present an analysis of 14 strains of ransomware that infect Windows platforms, and we do a comparison of Windows Application Programming Interface (API) calls made through ransomware processes with baselines of normal operating system behavior. The study identifies and reports salient features of ransomware as referred through the frequencies of API calls.

Keywords: Cryptovirology, cybersecurity, intrusion detection, malware, ransomware, Win/32

1. Introduction

Malware or malicious software is defined as any program or process that is crafted by the adversary to either affect routine operations of a computer, its operating system and hosted software, or to steal sensitive data. When such malware is crafted with the intent of extorting user data and holding it for ransom, then it is categorized as *ransomware*. While malware has been persistent for decades, the emergence of ransomware as the next big threat adopts a new business model by threat actors. The evolution of malware capabilities over the past 30 years is attributed to the rapid advances in computing power, memory, and communication bandwidth. Extortion of user data through malware dates back to 1989, when the PC CYBORG (AIDS) Trojan was released on floppy disks. Infected floppy disks when inserted by naïve users into their workstations would cause a Trojan infection, locking user files using basic cryptographic techniques, presenting a message stating the user's 'breach of software license', and demanding an amount of approximately US $200 for release of the extorted data. The Trojan was not very successful because the payment procedure adopted by the adversary was through bank cheques and the proliferation of malware through the crude floppy-disk medium was excruciatingly slow.

Strong data encryption techniques, attributed to advances in computing power and memory technology/affordability, alongside advances in payment techniques and cryptocurrency [1] have led to rapid evolution of ransomware during the period 2007-

2016. The ability of the adversary to conceal his/her identity and reaps profit through ransomware infections proliferating across billions of Internet-connected devices, is thus easily achievable in today's highly connected landscape. CTB-Locker (Curve, TOR, Bitcoin), is considered to be the first variant of ransomware to effectively combine three key characteristics required to achieve a high degree of success in infection, namely, the *anonymity capabilities* of the TOR routing protocol to conceal adversary location, the *anonymous payment capabilities* of Bitcoin to keep payment path untraceable, and *strong encryption* based on Elliptic Curve Cryptography with sufficient key lengths to resist attempts to crack the key including those involving brute-force [2].

In 2013, a 500% growth in ransomware variants and capabilities was reported [2]. This can be attributed to the three technological advances enumerated above. The common families of ransomware alongside their respective dates of emergence are listed as follows [2]: PC CYBORG Trojan (12/19/1989), One Half Virus (>1994), GPCode family (~2004), Reveton (~2012), CryptoLocker (~2013), CryptoWall (~2014), CryptoDefense (~2014), PoshCoder (~2014), Virlock (~2014), TeslaCrypt (~2015), CryptoFortress (~2015), CryptoTorLocker2015 (~2015), CTB-Locker (~2015), CryptoWall (~2016), Xorist (~2016), Filecoder (~2017) along with variants such as Petya (~2017), JAFF (~2017), and Wannacrypt (~2017).

Ransomware evolution witnessed the first brief increase in 2006-07 [3], mainly through the emergence of the GPCode variants. The GPCode.ak variant in particular was known to write the encrypted file contents to a new location in the user's disk, deleting the unencrypted user files. Through application of the '*undeletion utility*', partial recovery of user data was possible without having to pay the ransom to the adversary. Newer variants of GPCode used stronger encryption techniques with longer encryption keys (1024 or 2048 bits), thus encumbering the user data recovery attempts at the victim's machine.

A close look at the evolution of several versions of ransomware releases revealed that they were mostly copy-paste code from previous versions. Therefore, many of the limitations of one version were carried over to the next. In addition, several ransomware variants operated in unconventional ways. For instance, the Reveton ransomware [2], released in 2015, was found to merely lock the operation system's boot process without encrypting user data. Consequently, the ransomware activity was limited to disruption of operations and recovering user data without having to pay the ransom amount, was found to be easily achievable.

Another observed characteristic of recent ransomware traits is the ransomware procedural requirement to contact a centralized Command-and-Control (C2C) Server, once the victims' machine is infected, prior to encrypting the data. The C2C Server typically holds the cryptographic key required to decrypt the victim's data which has been held for ransom. In summary, the four stages of a ransomware-based attack can be described as follows:

- *Infection*: The ransomware software infects a victim's machine when the naïve victim opens an attachment that accompanies a spam message. Alternately, the victim's machine can also be infected when a compromised website is accessed.

- *Data encryption*: Once the victim's machine is infected with ransomware, cryptographic keys utilizing the Public Key Infrastructure (PKI) are generated either on the infected PC or the C2C server. The ransomware then proceeds to lock down the user's files or device. Ransomware specific definitions commonly result in one of two actions being undertaken: either the data/files on the victim's machine are attacked on a file-by-file basis, or critical filesystem structures such as the Windows Master File Table are altered. In both cases, the original files or data are encrypted with the host specific cryptographic keys, and the original files or metadata are then deleted.
- *Demand*: The ransomware software displays a message to the victim demanding that a certain amount be paid so as to release the locked data/files.
- *Outcome*: Based on the action taken by the victim, the following are possible outcomes: a) the data is recovered through elimination of ransomware trait from the victim's machine without paying the ransom amount, b) payments are made through anonymous channels such as BitCoin/MoneyPak or DarkCoin, or c) payments are not made and the ransomware trait is not eliminated, upon which the data/files are destroyed; with no backup in place, permanent loss of victim's data/files thus occurs.

It can be seen from the above examples that ransomware activity must by nature follow specific patterns of behavior. These patterns include the file identification process, encryption of files, network command and control communications, and use of anonymous networks. Quite simply, there is no optimal way to scan files and encrypt their content without making system level calls facilitated through the Windows Application Programming Interface (API). The Window API [4] provides a set of programming interfaces that simplify the process of developing software. For example, while a developer makes the system call "FileOpen", the operating system executes a series of instructions to locate the file in the file system, checks file access rights and permissions, and locates the file on the hard disk before returning the handle or reference back to the developer. By using the Windows API, developers are free to focus on the logic of their program (or malware) code and use the pre-defined procedures to accomplish their tasks [5].
Windows API sequence of calls has been an area of research during the recent past. In [6], the authors have presented a ransomware detection scheme that operates on Windows platforms and identifies modifications to various application types. Thirty most common Windows applications were evaluated and attempts by ransomware to access these file types, were analysed and reported.

. In [7], the authors present a call tracer approach for identifying the sequence of Windows API sequence of calls, by comparing the patterns of calls with known databases of malware, and by applying machine learning techniques for data analysis. Malware samples obtained from popular repositories were analysed and the results of the machine learning based classification of these samples were reported. In [8], the authors proposed an approach for identifying API sequence calls for malware samples. The lack of accuracy in anti-virus tools was highlighted as one of the motivations for the research conducted. Malware behaviour was generalised across 23,080 popular samples of malware.

As the number of Windows API calls is limited, and generally lower level file, network and cryptographic operations are exposed through a limited set of instructions, it may be possible to detect ransomware specific activities by analyzing their usage (or calls) to certain Windows API functions. We analyse and report ransomware activity based on the executing payload that has been transferred to a victim's machine beforehand. API call patterns and frequency analysis are used to help determine the behaviour of ransomware in a real-world environment. By identifying the programming patterns used by ransomware programmers, we can improve Operating System or Kernel level protection mechanisms. TBased on the results reported in this paper, we provide a fundamental platform for researchers to examine methods of ransomware detection based on behavioural analysis and/or entropy-based analysis, for future research.

2. Method, experimental setup, data processing and analysis

2.1 Method

We selected ransomware strains from recently circulated and well publicised ransomware variants [9][10][11] from various online resources. Ransomware strains were analysed based on their individual behaviour patterns. We tested ransomware and normal (non-malicious) baseline operations in successive experiments on a standardized Virtual Machine (VM). For each experimental test, we reset the VM to the same initial configuration, loaded a target test case, started the VM, logged Process Monitor events for a fixed duration of 10 minutes, and then halted the machine, and finally we exported the logged data and saved it for analysis. All ransomware tests were fully automated with the experimental tests execution and data collection scripted through a combination of BASH scripts, batch files and PowerShell scripts to ensure uniformity over each experimental test. For some baseline experimental tests the automated scripts were modified or customized for specific (often interactive) operations such as software installation and simulated web browsing.

2.2 Experimental setup

We created a 32-bit Windows 8 Virtual Machine in a Virtual Box. This Test VM was provided with a firewalled Internet connection through an intermediate VPN router and firewall. We preloaded the virtual disk image with a mix of documents, picture and video files. We saved them to multiple locations on the disk image. A Virtual Box shared folder was mapped to a drive letter and loaded with a multilevel directory structure, documents and media files. The virtual disk image and the shared folder were reset to initial conditions for each experimental test conducted. Table 1 presents the user file structure of the virtual machine deployed. The total disk space was 25 GB with 13.5 GB used space. Table 2 shows the count of the numbers of files in the shared network folder. Image files were the most popular file types whereas PDF files were found to be the least frequent.

Location	**File count and size**
Desktop	1.07GB, 442 files, 90 folders
Documents	524MB, 66 files, 22 folders

Pictures	417MB, 1344 files, 9 folders
Videos	661MB, 16 Files, 0 Folders

Table 1 - Virtual machine victim user's file structure

File type	**Count of files in shared (network) folder**
jpg and png image files	1337
ppt (and pptx)	2
pdf	55
doc (and docx)	34
xls (and xlsx)	17
mp3/mp4 (audio and video media)	20
other filetypes	27
directory and subdirectory entries (maxdepth = 5)	31

Table 2 - Shared folders (network) file counts

Automation element/method	**Automated on Host or Guest**	**Description**
Create PowerShell experiment execution control file	Host	Create a control file for use by Windows PowerShell on boot up – specifies the experiment to run, working directory and various other conditions
VirtualBox automation	Host	Use of BASH script and VirtualBox Manage commands to start/stop and reset VM snapshots
PowerShell scripts	Guest	Read the control file from a shared read-only resource to identify experiment's executable and run conditions
Shared filesystem reset	Host	BASH script to reset shared filesystem resources
Timed hard shutdown	Host	Hard shutdown of the guest VM and restart into data extraction mode
Data extraction	Guest and Host	Restart and launch ProcMon to recover boot time data. Save files and then move extracted files to a safe location for future analysis

Table 3 - Technology and elements used for automation of testing

In Table 3, we define various automation procedures that were executed during the experiments.

2.3 Data processing and analysis

We loaded the Process Monitor's data in a clean Windows 10 Virtual Machine, and then re-exported it with complete stack traces and Windows debugging symbols to XML format.

We processed the exported Process Monitor's event data using a Python script to extract each stack trace, and examined each call frame in order to identify the first call to a Windows system file. A sample contingency table is provided in Table 4. The calling address, the resolved called symbol, the Dynamic Link Library (DLL) path and metadata were extracted. The extracted API data consisting of 36 million Windows API calls across all 30 experimental tests were further summarized into two-way contingency tables plotting Windows API Call frequencies for each API Call and experimental test combination.

System/API Call	**CTB-Locker_A.csv**	**Revenge_A.csv**	**…**	**Explorer Session**	**Install MS Office**	**Run Powerpoint + User Activity**
CloseHandle	4	10946	**…**	*944*	0	1545
CoCreateInstance	70	48	**…**	*4567*	0	0
CoInitialize	44	0	**…**	289	542	0
CoInitializeSecurity	38	0	**…**	*116*	0	0
…	…	…	**…**	*…*	…	…
(Total of 1262 Calls)						

Table 4 - Sample contingency table comprising 1262 rows and 25 columns

	Experimental test	**File Hash (SHA 256)**
Baseline operations	Baseline Boot to Idle	NOT RECORDED
	Windows Explorer Session Navigating OS and Folders	NOT RECORDED
	Install MS Office 07	NOT RECORDED
	Install Kodi Media Centre	NOT RECORDED
	Installing Firefox Browser	NOT RECORDED
	Installing Apache Open Office	NOT RECORDED
	Installing Logitech Media Centre	NOT RECORDED
	Running Word	NOT RECORDED
	Running PowerPoint	NOT RECORDED
	Running Excel	NOT RECORDED
	Running Apache Open Office	NOT RECORDED
	Internet Browsing in IE	NOT RECORDED
	Running Firefox Browser	NOT RECORDED
	Run Kodi Media Centre	NOT RECORDED
	Run Logitech Media Centre Control	NOT RECORDED
Ransomware tests	CTB-Locker	128a0f0cd5d10f864d5a0741ba25996b2bf74f580ac7918dec6516215801e39a

	Experimental test	File Hash (SHA 256)
	Cerber	cf262a9236eaf5230c219845823f36fd8c8e8b77ba882c34ce38a5087539cf71
	CrypMIC	b2bcfc4c5d1d60f7ea4298d32dcfff303f4db4b1ba89a8b6d24b7ccfe883e45a
	CryptFile2	a1e4693db6419eb5588f25d2b9f90db6c0e96e30a51fed5f0236cbdd49894e75
	CryptoMix	a9a232cbff2c4347c1fcdeb1a3f1a6e45fbd4e93a107c6dd57fb8994df9d3bce
	CryptoShield	d56fb2bdad7a50ab1f6ef76c67669452ed4da2bf865beafcf4956ab30bfa20fc
	GlobeImposter	72ddceebe717992c1486a2d5a5e9e20ad331a98a146d2976c943c983e088f66b
	Gryphon	933af0c69e1e622e5677e52c24545761c2843b3f52ea38e63bbe4786bfd6276e
	JAFF	824901dd0b1660f00c3406cb888118c8a10f66e3258b5020f7ea289434618b13
	Mole	c2e1770241fcc4b5c889fec68df024a6838e63e603f093715e3b468f9f31f67a
	NemucodAES	482711b2f17870ddae316619ba2f487641e35ac4c099ae7e0ff4becd79e89faf (payload)
	Revenge	8ab65ceef6b8a5d2d0c0fb3ddbe1c1756b5c224bafc8065c161424d63937721c
	TeslaCrypt	200bc25fa093ce65f41baa1c3efe02dcc238b04cb57a6fc5ee87da1e04d6e168
	WannaCry	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa

Table 5 - Experimental tests - baseline and ransomware (hashes where applicable)

In Table 5 shows a list of all experiments, baseline and 14 ransomware strains, that were conducted. Table 6 describes the baseline experiments that we executed. In particular, we identify the system activities and/or events that were executed during each Win/32 baseline operation.

Baseline operation	Experiment fully automated? [Yes/No]	Description of system events, activities
Baseline boot to idle	Yes	Standard windows housekeeping, connection to network shares, basic operations
Windows Explorer session navigating operating system and folders	No (opened one folder every 30 seconds, copied folder from network share to desktop, created folder ever minute, moved files from desktop to created folders, deleted a folder every 2 minutes)	Common user activity. Opening directories and locations, listing files, querying file types to identify associated applications
Install MS Office 07	Yes	Writes many files, registers components and COM objects
Install Kodi media centre	Yes	Non-Microsoft application handles many media file types
Installing Firefox browser	Yes	Non-Microsoft application handles internet and registers application capable of using Internet. Imports/reads data from other browsers

Baseline operation	Experiment fully automated? [Yes/No]	Description of system events, activities
Installing Apache open office	No	Alternate to Microsoft Office examine the difference in coding standards for similar activities
Installing Logitech media centre	No	Media server. registers and starts services and network listeners
Running word	No (Launch automated by script – Activity manual: opened one document every 2 minutes, copy and pasted text every 30 seconds, typed for 1 minutes, imported media, ran spell check and saved document)	Common user activity
Running PowerPoint	No (Launch automated by script – Activity manual: opened one document every 2 minutes, copied and pasted text every 30 seconds, typed for 1 minute, imported media, ran spellcheck and saved document)	Common user activity
Running Excel	No (Launch automated by script – activity manual: opened one document every 2 minutes, created sheet created 100 rows of data, created chart)	Common user activity
Running Apache open office	No (Initial launch automated by script – activity manual: rotated through test sequences for Word, PowerPoint and Excel until time expired)	Common user activity provides alternate to Microsoft office application. Examine the effect of different coding standards on similar activities to Running Microsoft Office Products
Internet browsing in IE	No (Launch automated by script – activity manual: browsed to a list of pages, one page every 30 seconds)	Common user activity uses network
Running Firefox browser	No (Launch automated by script – activity manual: browsed to a list of pages, one page every 30 seconds)	Common user activity uses network. Alternate coding standards for similar activities as Internet Browsing in IE
Run Kodi media centre	No (Launch automated by script – allowed to complete normal start up media scans – further manual activity: browse a media folder every minute, play one video file, added a new media folder and scanned for content)	3rd party developed, accesses many files, catalogues media, connects to the internet to identify media. Scans specified file system locations for media files
Run Logitech media centre control	No (Required admin privileges and launch from control panels. Added a new media folder and scanned for content.	Launches control panel, allows users to identify media folders and execute file system scans of specified locations for media files. Starts and stops the media centre service

Table 6 – Baseline operations and description of system events and activities

3. Analysis

Of the 1262 calls to external functions across all experiments, 244 were present in ransomware which were further reduced to 209 calls by combining similar calls of ANSI and Unicode variants as shown in Table 7.

Calls	Grouped into
CopyFileA CopyFileExW CopyFileW	CopyFile [A|ExW|W]
CreateDirectoryA CreateDirectoryW	CreateDirectory [A|W]
…A …W …Ex …ExA …ExW	…[A|W|Ex|ExA|ExW]

Table 7 - Merging of similar API Calls

The rationale for merging similar API calls is that Windows API calls such as FindNextFileW and FindNextFileA are essentially the same API call (the 'W' variant accepting Unicode and 'A' variant accepting ANSI coded input strings). Similarly, functions with Ex suffixes are generally newer with a different call pattern, however their base functionality is often quite similar.

The API calls were arranged into two-way contingency tables that plotted the observed frequency of each API call for each experimental test. We identified API *calls of interest.* Calls of interest were selected where the "API call's presence indicated ransomware regardless of call frequency" and "API calls with significantly higher-than-average call frequencies" statistics. We used Fisher exact tests to compare the prevalence of each specific API call in the ransomware group to the normal baseline operations group. Calls with usage patterns that differed significantly ($p < 0.05$) between the two groups were identified.

3.1 Results

An initial examination of the contingency table that compares all ransomware system calls to system calls made by non-malicious normal baseline operations show that ransomware used a small subset of all system calls logged during normal baseline operations. Comparing the frequency of all ransomware system calls to the frequency of system-calls in normal baseline operations shows that identification of ransomware can be done through call frequencies alone (chi-square; $p << 0.01$; 95% confidence level for significance testing). This is a reasonable expectation given the large data set and high variability in call frequencies and prevalence. The API calls which contributed most to the chi-square statistic were examined to determine what subset of calls could be used to indicate the presence of ransomware activity.

When we examine individual API calls more closely, we found that 18 Windows API calls where usage patterns (prevalence or call frequency) varied between ransomware and baseline normal operation differed significantly (Tables 8, 9 and 10). These API calls occur in significantly more ransomware strains (compared to baseline experiments), or at greater call frequencies ($p < 0.05$).

The interesting calls identified included:

- 8 API calls that existed only in ransomware at a significant level.
- 4 API calls that existed in both ransomware and normal operations, where the difference in utilization of the API call was statistically significant and more common in ransomware samples than in normal baseline operations.
- 6 API calls that existed in both ransomware and baseline normal operation and where the ransomware frequency count exceeded the baseline mean by more than three standard deviations (3σ).

	Windows API Call	**Count of ransomware samples used**	**Count of baseline samples used**	**Usage differs between ransomware and baseline (Fisher exact P-value)**
Present only in ransomware	InternetOpen	6	0	0.006
	CryptDeriveKey	5	0	0.017
	CryptDecodeObject	4	0	0.042
	CryptGenKey	4	0	0.042
	CryptImportPublicKeyInfo	4	0	0.042
	GetUserName	4	0	0.042
	NdrClientCall2	4	0	0.042
	socket	4	0	0.042
Used in more ransomware strains	_tailMerge_CRYPTSP_dll*	9	1	0.002
	CoCreateInstance	8	1	0.005
	SHWindowsPolicy	8	1	0.005
	GetFileType	10	4	0.027

Table 8 - Calls to Windows APIs (without considering call frequency) - ransomware vs normal baseline operations

	Windows API Call	**Count of ransomware samples using high ($\bar{x} + 3\sigma$) frequency calls rates**	**Count of baseline samples using high ($\bar{x} + 3\sigma$) frequency call rates**	**Significance (Fisher exact)**
Used in ransomware at higher call frequency	CryptAcquireContext	7	0	0.002
	CloseHandle	6	0	0.006
	FindNextFile	6	0	0.006
	SetFilePointer	6	1	0.035
	GetFileSize	4	0	0.042
	SetFileAttributes	4	0	0.042

Table 9 - Calls to Windows APIs where ransomware call frequency exceeds baseline mean call frequency by more than 3 standard deviations.

	Windows System Call	CTB-Locker	Cerber	CrypMIC	CryptFile2	CryptoMix	CryptoShield	GlobeImposter	Gryphon	JAFF	Mole	Revenge	*TeslaCrypt*	WannaCry	NemucodAES
Detected by call presence (exclusive to ransomware)	InternetOpen				*	*	*			*	*	*			
	CryptDeriveKey				*	*	*		*			*			
	CryptDecodeObject		*						*		*	*			
	CryptGenKey						*			*	*	*			
	CryptImportPublicKeyInfo		*						*		*	*			
	GetUserName				*		*				*	*			
	NdrClientCall2					*	*		*			*			
	socket		*	*		*	*								
Detected by call presence (not ransomware exclusive)	_tailMerge_CRYPTSP_dll (1 false positive)		*	*		*	*	*	*	*		*		*	
	CoCreateInstance (1 false positive)	*	*			*	*		*	*		*			*
	SHWindowsPolicy (4 false positivies)		*		*	*	*		*	*		*			*
	GetFileType (1 false positive)	*	*		*	*	*	*	*	*		*			*
Detected in ransomware through statistically high ($\bar{x}+3\sigma$) call frequencies	CryptAcquireContext				*	*	*	*	*		*	*			
	CloseHandle			*		*	*	*	*			*			
	FindNextFile			*			*	*	*	*		*			
	SetFilePointer (1 false positive)		*		*	*	*	*				*			
	GetFileSize			*				*	*	*					
	SetFileAttributes		*				*				*	*			
	Count of calls capable of identifying ransomware	2	9	5	7	11	15	7	12	8	7	16	*0*	1	3
	* - Ransomware Detected with System Call														

Table 10 - Calls to Windows APIs categorized by ransomware strain.

The fisher-exact test of independence showed a very high level of certainty that the baseline versus ransomware samples differed through a systematic process, namely, that the presence of ransomware in the system and not in our baseline tests was not merely coincidental. For example, GetFileType is used more often in ransomware than in baseline samples runs (10 ransomware samples vs 4 baseline operations). However, due to the small sample sizes for both baseline and ransomware, the difference in the API usage by ransomware strains and the baseline tests within the significant range ($p=0.066 > 0.05$). As such, no specific API can be used for detecting ransomware. Rather, the APIs identified and reported in Table 10 can aid in the detection of ransomware strains that would otherwise remain undetected in a Win/32 standard operating environment. It must also be noted that none of these APIs are dangerous for a standard Win/32 operating environment. However, based on our findings, we found that calls to some of these APIs are more frequent than others during a ransomware infection.

4. Discussion

Ransomware activities were clearly identified in thirteen out of fourteen ransomware strains using Windows API calls of interest. Of these, nine were identified calls that were unique to ransomware and did not trigger false positive events during detection. Only the variant of TeslaCrypt tested was not identified.

One third of the API calls of interest were related to cryptographic activities. These calls were primarily used to obtain handles to key containers [12] and generate public and private keys. The presence of cryptographic API calls is reasonable and expected for typical ransomware activities. A delayed load tailMerge of CRYPTSP.DLL [13] was also present in nine strains of ransomware. This cryptographic service provider dynamic link library appears to be a legacy crypto library the use of which has been identified and discussed in [14].

A further six of the API calls were related to filesystem operations. These included calls to scan directory structures for files, examine file types, sizes and set pointers to allow the ransomware to read and write file contents. Five out of six file operations were detected through API call frequency analysis indicating that while non-malicious activities also resulted in file activity, the rates observed in ransomware significantly exceeded ($\bar{x}+3\sigma$) the call rates during normal system operations.

Internet and socket connections were surprisingly absent from normal non-malicious operations. It appears that the coding patterns employed by windows and open-source software developers do not often create direct network sockets or connections. Socket operations tend to be low-level in nature, which means coding complete network protocols using sockets is likely to be a laborious task. Socket programming is useful for limited and specific tasks that require lightweight network listeners or clients with well-defined communications protocols [15][16].

Four ransomware strains utilized NdrClientCall2 which is associated with the Windows Remote Procedure Call (RPC) interface. RPC is used to create client server applications without the need to manage the underlying network protocols and communications [17].

For example, RPC could allow ransomware developers to establish command and control server communication without resorting to socket level programming.

CoCreateInstance was used non-exclusively in eight out of the fourteen ransomware strains. CoCreateInstance appears to be used by ransomware to access Windows COM objects through unique class and instance identifiers. Developers may use CoCreateInstance to obtain access to a COM handler instance that can perform a wide range of windows actions including creating file links, spawning shells and scheduling start-up items. While this is a perfectly legitimate programming technique, it appears to be rare among legitimate baseline samples. It has also been observed in ransomware examples to obfuscate code being executed and provide a mechanism to bypass Microsoft's Antimalware Scan Interface [18][19].

5. Conclusion

In this work, we have successfully identified Windows API calls that differ significantly in their usage between normal non-malicious operations and ransomware activities. These low-level system calls may be useful in identifying ransomware without specifically identifying code signatures within the ransomware executable. The goal of this research was to investigate API calls that could allude toward ransomware infection. Based on the findings reported in this paper, we can have a now better understanding of what the ransomware strain is actually doing on the system in terms of API calls. Our research results obtained in this work will help in the future development of better anti-virus software, additional security controls including Intrusion Detection System (IDS), or even in hardening kernels by allowing them to detect multiple API calls.

Given the nature of many of the identified Windows API calls, detection of ransomware activity may be possible at the operating system level. Our research found several API calls of interest that were predominantly present in ransomware. By further combining the detection of these API calls it would be possible to further reduce the false positive rate and increase the detection rate.

As the Windows APIs that have been discussed operate as low-level calls of the operating system, we expect that circumventing detection by using different APIs will be a complex process requiring developers to statically link complex file system and network code into their malware binaries. This type of code embedding would greatly increase the size of malicious executable and would funnel API calls to even lower levels. We believe this approach is unlikely to be successful for ransomware developers because the execution of code that directly calls low level drivers and file system APIs is uncommon and would lead to easy detection.

References

[1] Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S. Bitcoin and

Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press, 2016.

[2] Hampton, N., Baig, Z., Ransomware: Emergence of the Cyber-Extortion Menace, In Proc. Of the Australian Information Security Management Conference, Perth, Australia, 2015.

[3] Luo, X., Liao, Q., "Ransomware: A new cyber hijacking threat to enterprises," Handbook of Research on Information Security and Assurance, 2009.

[4] Marhusin, M., Larkin, H., Lokan, C., Conrnforth, D., An Evaluation of API Calls Hooking Performance, In Proc. of the Intl' Conf. on Computational Intelligence and Security, Suzhou, China, 2008.

[5] Shankarapani, M., Ramamoorthy, S., Movva, R., Mukkamala, S., "Malware detection using assembly and API call sequences," Journal in Computer Virology, 7(2), 2011.

[6] Scaife, N., Carter, H., Traynor, P., Butler, K., "Crytodrop (and Drop It): stopping ransomware attacks on user data," IEEE 36th Intl' Conf. on Distributed Computer Systems, 2016.

[7] Ravi, C. and Mahoharan, R., "Malware Detection using Windows Api Sequence and Machine Learning," Intl. Jour. of Computer Applications, 43(17), 2012.

[8] Ki, Y., Kim, E., Kim, H., "A Novel Approach to Detect Malware based on API Call Sequence Analysis," Intl. Jour. of Distributed Sensor Networks, 2015.

[9] Krishnan, R., CTB-Locker Ransomware Spreading Rapidly, Infects Thousands of Web Servers, https://thehackernews.com/2016/02/ctb-locker-ransomware.html, Last Accessed: 17th November, 2017.

[10] Abrams, L., TeslaCrypt shuts down and Releases Master Decryption Key, https://www.bleepingcomputer.com/news/security/teslacrypt-shuts-down-and-releases-master-decryption-key/ Last Accessed: 17 November, 2017.

[11] Mohurla, S. and Patil, M., "A brief study of Wannacry threat: ransomware attack 2017," Intl' Jour. of Advanced Research in Computer Science, 8 (5), 2017.

[12] CryptAcquireContext function, https://msdn.microsoft.com/en-us/library/windows/desktop/aa379886(v=vs.85).aspx, , Last Accessed: 17 November, 2017.

[13] Pietrek, M., Under The Hood, https://www.microsoft.com/msj/1298/hood/hood1298.aspx, Last Accessed: 17 November, 2017.

[14] Palisse, A., Bouder, H., Lanet, J-L., Guernic, C., Legay, A., Ransomware and

the legacy Crypto API, Lecture Notes in Computer Science 10158, pp. 11-28, 2017.

[15] Avoiding Common Networking Mistakes, https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/CommonPitfalls/CommonPitfalls.html, Last Accessed: 17 November, 2017.

[16] Windows Apps Team, Networking API Improvements in Windows 10, https://blogs.windows.com/buildingapps/2015/07/02/networking-api-improvements-in-windows-10/, Last Accessed: 17 November, 2017.

[17] Remote Procedure Call, https://msdn.microsoft.com/en-us/library/windows/desktop/aa378651(v=vs.85).aspx, Last Accessed: 17 November, 2017.

[18] Antimalware Scan Interface, https://msdn.microsoft.com/en-us/library/windows/desktop/dn889587(v=vs.85).aspx, Last Accessed: 17 November, 2017.

[19] Antimalware Scan Interface, https://enigma0x3.net/2017/07/19/bypassing-amsi-via-com-server-hijacking/, Last Accessed: 17 November, 2017.