Static and Dynamic Analysis of WannaCry Ransomware

Maxat Akbanov*, Vassilios G. Vassilakis*, Ioannis D. Moscholios†, Michael D. Logothetis‡
* Dept. of Computer Science, University of York, York, United Kingdom
† Dept. of Informatics & Telecommunications, University of Peloponnese, Tripolis, Greece
‡ Dept. of Electrical & Computer Engineering, University of Patras, Patras, Greece

***Abstract*—Nowadays ransomware presents a huge and the fastest growing problem for all types of users from small households to large corporations and government bodies. Modern day ransomware families implement sophisticated encryption and propagation schemes, thus limiting chances to recover the data almost to zero. In order to design and develop appropriate detection and mitigation mechanisms it is important to perform ransomware analysis and indemnify its features. In this work, we present our ransomware analysis results focusing on the infamous WannaCry ransomware. In particular, the presented research examines the WannaCry behaviour during its execution in a purpose-built virtual lab environment. We perform static and dynamic analysis using a wide range of malware analysis tools. The obtained results can be used for developing appropriate detection and mitigation mechanisms for WannaCry or other ransomware families that exhibit similar behaviour.**

Keywords*—*Malware analysis, ransomware, WannaCry

I. Introduction

Currently ransomware threat is considered as the main moneymaking scheme for cyber criminals and the key threat to the Internet users [1], [2]. Starting from relatively simple fake antivirus applications in 2008, ransomware has evolved during the time and emerged into sophisticated forms such as crypto type ransomware. The apotheosis of this evolution is the occurrence of a new type of ransomware which combines the usage of exploits with worm-like spreading mechanisms to propagate itself in both internal and external networks. Moreover, the emergence of new types of ransomware, such as WannaCry, showed that ransomware keeps evolving and cyber criminals are upgrading the ransomware code with more sophisticated features, such as worm propagation components and public-key encryption mechanisms. Therefore, from the research perspective, the design of new countermeasures apart from traditional security approaches, is considered as important and trending task in this field. Such designs, however, require a comprehensive analysis of ransomware features and behaviour which typically involve a wide range malware analysis tools.

In this work, we have performed a comprehensive analysis of the infamous WannaCry ransomware. We present both static and dynamic analysis results. The presented techniques are applicable also in the cases of other ransomware families with characteristics similar to WannaCry, such as worm-spreading mechanisms and public-key based encryption. In particular, the presented research examines the WannaCry behaviour during its execution in a safe purpose-built virtual lab environment at the University of York. The obtained results can be used for designing and developing effective ransomware detection and mitigation mechanisms.

The rest of paper is organized as follows. In Section II, we present the relevant background information on ransomware in general and on WannaCry in particular. In Sections III, IV, and V, we present the main findings from our conducted static and dynamic analysis of WannaCry, including its inherent network indicators. Finally, Section VI draws the conclusions and discusses potential future directions.

II. Background

A. The Basics of Ransomware

Ransomware presents a type of malicious software that prevents or limits users from accessing their system, either by locking the screen or by encrypting files, until a ransom is paid [3]. Typically, two types of ransomware are distinguished: *lockers* and *cryptors* [2]. Lockers present a less sophisticated type of ransomware which simply locks the device's user interface, preventing from logging in and accessing programs and data. In most cases it leaves the user with very few capabilities such as allowing the victim just to communicate with the attacker and pay the ransom. Lockers usually can be removed cleanly, as they leave the underlying system and files untouched. This makes lockers less effective at extracting ransom payments compared with their more destructive relatives - cryptors.

On the other hand, cryptors represent an advanced type of ransomware which aims at encrypting specific files of the infected system. Cryptors use a variety of different cryptographic algorithms, including both symmetric and public-key based. Cryptors that rely on public-key encryption are particularly difficult to mitigate, since the encryption keys are stored in a remote command and control (C&C) server. Cryptors typically include a time limit for ransom to be paid and provide users with a special website to purchase cryptocurrency (e.g., Bitcoins) and step-by-step instructions on how to pay the ransom. The lifecycle of modern day ransomware typically consists of the following steps [4]: distribution, infection, communications, file search, file encryption, and ransom demand.

B. The Basics of WannaCry

WannaCry ransomware (also known as Wana Decrypt0r, WCry, WannaCry, WannaCrypt, and WanaCrypt0r) was observed during a massive attack across multiple countries on 12 May 2017 [5]. According to the multiple reports from

security vendors, in total 300 000 systems in over 150 countries had been severely damaged. The attack affected a wide range of sectors, including healthcare, government, telecommunications, and gas/oil production.

A difficulty of protecting against WannaCry lies in its ability to spread itself to other systems by using a worm component. This feature makes the attacks more effective and requires defense mechanisms that can react quickly and in real time. Furthermore, WannaCry has an encryption component that is based on public-key cryptography.

During the infection phase, WannaCry uses the *EternalBlue* and *DoublePulsar* exploits, that were allegedly leaked in April 2017 by a group called The Shadow Brokers. EternalBlue exploits the server message block (SMB) vulnerability that was patched by Microsoft on March 14, 2017 and has been described in the security bulletin MS17-010 [6]. This vulnerability allows the adversaries to execute remote code on the infected machines by sending specially crafted messages to an SMBv1 server, connecting to TCP ports 139 and 445 of unpatched Windows systems. In particular, this vulnerability affects all unpatched Windows versions starting from Windows XP to Windows 8.1, except for Windows 10.

DoublePulsar is a persistent backdoor that can be used to access and execute code on previously compromised systems, thus allowing the attackers to install additional malware on the system. During the distribution process, WannaCry's worm component uses the EternalBlue for the initial infection through the SMB vulnerability by actively probing appropriate TCP ports and if successful, tries to implant the DoublePulsar backdoor on the infected systems.

III. WannaCry Static Analysis

In this section, we present our findings from static analysis of WannaCry. To perform the analysis, two virtual machines (VMs) were used. The characteristics of the host machine are: Intel Core i7-4700MQ 2.40 GHz and 16 GB RAM. The 1st VM was running Windows 7 SP1 and was infected with WannaCry. The 2nd VM was running REMnux [7], which is a free Linux toolkit for reverse-engineering and malware analysis.

Samples of WannaCry were obtained from VirusShare [8]. In particular, we analyzed two executable files: the worm component and the encryption component (Table I). Below we describe our main findings.

Analysis with Pestudio tool [9] has revealed that the worm and the encryption components contain dynamic-link libraries (DLLs), as shown in Tables II and III. During its execution, the worm invokes *iphlpapi.dll* in order to retrieve network configuration settings for the infected host. The *kernel32.dll* and *msvcrt.dll* are two most invoked libraries by the encryption component. This may indicate that the main WannaCry encryption functionality was implemented by these two libraries. To confirm this, the imported functions of the libraries were observed with Pestudio. As it is shown in Tables IV and V, in general WannaCry uses Microsoft's crypto, file management and C runtime file application programming interfaces (APIs). The Crypto API library is used to generate and manage random symmetric and asymmetric cryptographic keys.

TABLE I. WannaCry components.

	Worm Component
MD5	db349b97c37d22f5ea1d1841e3c89eb4
SHA1	e889544aff85ffaf8b0d0da705105dee7c97fe26
SHA256	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
File Type	PE32 executable (GUI) Intel 80386, for MSWindows
	Encryption Component
MD5	84c82835a5d21bbcf75a61706d8ab549
SHA1	5ff465afaabcbf0150d1a3ab2c2e74f3a4426467
SHA256	ed01ebfbc9eb5bbea545af4d01bf5f107166184048043 9c6e5babe8e080e41aa
File Type	PE32 executable (GUI) Intel 80386, for MSWindows

TABLE II. DLLs invoked by WannaCry worm component.

Library	**Imports**	**Description**
ws2_32.dll	3	Windows Socket 2.0 32-bit
iphlpapi.dll	2	IP Helper API
wininet.dll	3	Internet Extensions for Win32
kernel32.dll	32	Windows NT BASE API Client
advapi32.dll	11	Advanced Windows 32 Base API
msvcp60.dll	2	Windows NT C++ Runtime Library
msvcrt.dll	28	Windows NT CRT

TABLE III. DLLs invoked by WannaCry encryption component.

Library	**Imports**	**Description**
kernel32.dll	54	Windows NT BASE API Client
advapi32.dll	10	Advanced Windows 32 Base API
user32.dll	1	Multi-UserWindows USER API Client
msvcrt.dll	49	Windows NT CRT

TABLE IV. WannaCry worm component functions.

Function	**Location**
GetCurrentThread	0xa53a
GetStartupInfoA	0xa97a
StartServiceCtrDispatcherA	0xa6f6
RegisterServiceCtrDispatcherA	0xa6d8
CreateServiceA	0xa688
StartServiceA	0xa662
CryptGenRandom	0xa650
CryptAcquireContextA	0xa638
OpenServiceA	0xa714
GetAdaptersInfo	0xa792
InternetOpenUrlA	0xa7c8

TABLE V. WannaCry encryption component functions.

Function	**Location**
OpenMutexA	0xda84
GetComputerNameW	0xd8b2
CreateServiceA	0xdc2a
OpenServiceA	0xdc62
StartServiceA	0xdc52
CryptReleaseContext	0xdc14
RegCreateKeyW	0xdc04
fopen	0xdcd4
fread	0xdccc
fwrite	0xdcc2
fclose	0xdcb8
CreateFileA	0xd922
ReadFile	0xd964

IV. WannaCry Dynamic Analysis

In this section, we present our findings from dynamic analysis of WannaCry. To this end, a virtual testbed of Fig. 1 was built. In particular, a custom network VMnet 5 - 192.168.180.0/24 was created with the Virtual Network Editor option in VMWare hypervisor. This scheme allows observing domain name system (DNS) queries made by WannaCry during the infection and replication process, as performed by the worm component across the internal and external networks via port 445 of SMBv1 protocol. The REMnux machine acts as DNS and HTTP server, and is able to intercept all network communications using Wireshark. DNS and HTTP services in REMnux were enabled using the FakeDNS and HTTP Daemon utilities, respectively.

Our dynamic analysis has revealed that, upon startup, the worm component tries to connect to the following domain, using the *InernetOpenUrl* function:

www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

The aforementioned domain is a kill-switch domain. That is, if the domain is active, the worm component stops running. On the other hand, if the worm component cannot establish a connection with this domain (e.g., if the domain is not active or if there is no connectivity), it continues to run and registers itself as a "Microsoft Security Center (2.0) Service" *mssecsvs 2.0* process on the infected machine.

The FakeDNS utility at REMnux captures the malicious DNS request on port 80 (Fig. 2), while Wireshark shows (Fig. 3) the DNS packet query field from the infected machine (IP 192.168.180.130) to the DNS server on REMnux (IP 192.168.180.128).

After installing itself as a service, the worm component extracts the hardcoded *R resource* and then copies it to *C:\Windows\taskche.exe*. The R resource represents the binary of the WannaCry encryption component. During its execution the encryption component checks if one of the following mutual exclusion objects (mutexes) exists:

$$GlobalnMsWinZonesCacheCounterMutexA$$
$$GlobalnMsWinZonesCacheCounterMutexW$$
$$MsWinZonesCacheCounterMutexA$$

If the mutex is present on the system, then the encryption component automatically stops without taking any further actions. Otherwise, the encryption process starts. To encrypt each file, a different 16-byte symmetric AES key is generated using the *CryptGenRandom* function. Then, every generated AES key is encrypted with the public RSA key (which is part of the encryption component) and stored inside the file header starting with *WANACRY!* string value. Encrypted files are renamed and added with the *.WNCRY* file extension. The encryption component contains a password-protected ZIP archive. We managed to obtain the password, "WNcry@2ol7", by disassembling the encrypter with the IDA Pro tool [10] (see Fig. 4). The contents of the ZIP archive are summarized in Table VI and described below:

- *msg* is a folder that contains a list of rich text format (RTF) files with *wnry* extension. These files are the readme instructions used to show the extortion message to the victim in different languages, based on the information obtained from the system by malicious WannaCry functions.
- *b.wnry* is an image file used for displaying instructions for the decryption of user files. It starts with 42 4D strings, which indicates that this file is a bitmap image.
- *c.wnry* contains a list of Tor addresses with *.onion* extension and a link to a zipped installation file of the Tor browser from Tor Project [11].
- *r.wnry* is a text file in English with additional decryption instructions to be used by the decryption component (the *u.wnry* file mentioned below).
- *s.wnry* file is a ZIP archive (HEX signature 50 4B 03 04) which contains the Tor software executable. This executable has been obtained with the assistance of the WinHex tool [12] by saving raw binary data with *.zip* extension.
- *t.wnry* is an encrypted file with *WANACRY!* encryption format. The file header starts with the *WANACRY!* string.
- *taskdl.exe* is a supporting tool for the deletion of files with *.WNCRY* extension. By observing the properties of the file, the following masquerade description can be found: "SQL Client Configuration Utility".
- *taskse.exe* is a supporting tool for malware execution on remote desktop protocol (RDP) sessions. The following file description was identified:" waitfor - wait/send a signal over a network".
- *u.wnry* is an executable file (HEX signature 4D 5A) with name "@WanaDecryptor@.exe", which represents the decryption component of WannaCry.

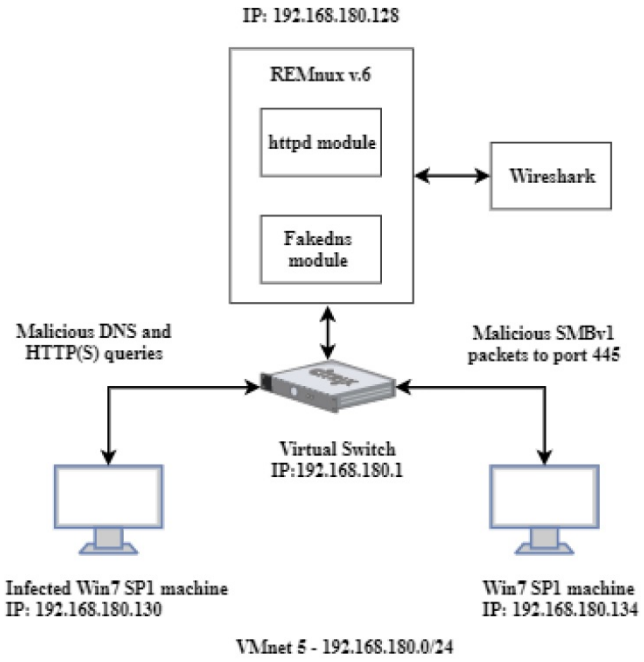

Fig. 1. Testbed for dynamic WannaCry analysis.

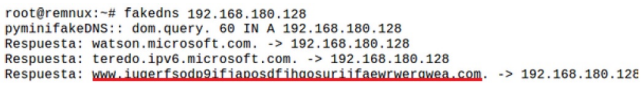

Fig. 2. FakeDNS capture of the malicious DNS request.

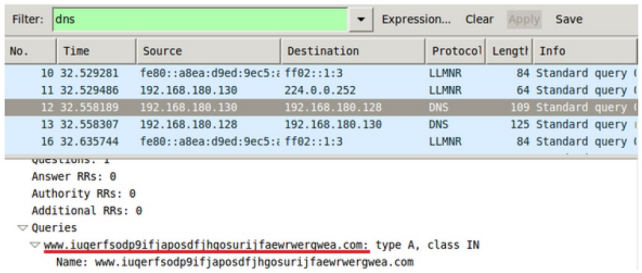

Fig. 3. Wireshark capture of the malicious DNS request.

Our dynamic analysis has also revealed that, to achieve persistence on the infected machine, WannaCry performs the following actions:

- Creates an entry in the Windows registry to ensure that it executes every time the machine is restarted.
- Attempts to achieve memory persistence by adding itself to the AutoRun feature of Windows.
- Uses Windows "icacls" command to grant itself a full access to all files on the machine.
- Deletes all backup (shadow) copies and tries to prevent being booted in *safe mode* by executing several commands in Windows command line.
- Deletes all backup catalogs.
- By using Winsdows command line, creates a VBScript program which generates a single shortcut of *WanaDecryptor@.exe* decrypter file.
- Tries to kill SQL and MS Exchange database processes by executing several commands in Windows command line.

V. WannaCry Communications

After performing initial interactions and checking connectivity with the kill-switch domain, the worm functionality is established by initiating the *mssecsvs 2.0* service. This service tries to spread WannaCry's payload through the SMB vulnerability on any vulnerable system.

In order to perform this, WannaCry creates two separate threads that simultaneously replicate the payload in internal (local) and external networks. In the internal network, before starting the propagation process, the worm component obtains the IP addresses of local network interfaces by invoking the *GetAdaptersInfo* function and determining the existing subnets.

Fig. 4. Password for a ZIP archive in the encryption component.

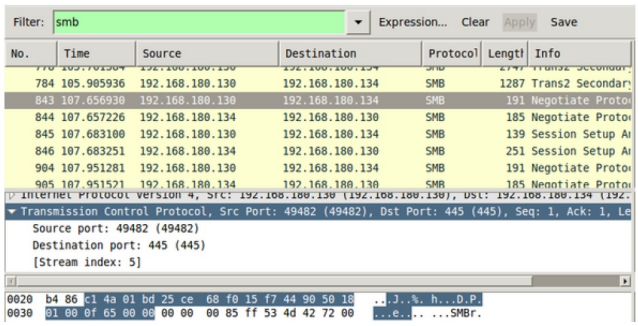

Fig. 5. WannaCry internal network traffic attempting the SMB exploit.

TABLE VI. Files in the password protected ZIP archive.

Name	Size (bytes)	Modified
msg	1,329,657	2017-05-11
b.wnry	1,440,054	2017-05-11
c.wnry	780	2017-05-11
r.wnry	864	2017-05-10
s.wnry	3,038,286	2017-05-09
t.wnry	65,816	2017-05-11
taskdl.exe	20,480	2017-05-11
taskse.exe	20,480	2017-05-11
u.wnry	245,760	2017-05-11

After that, the worm component tries to connect to all possible IP addresses in any available local network, on TCP port 445 (the default port for SMB over IP service). If successful, the worm component attempts to exploit the service for vulnerability described in MS17-010 [6]. During our experiments, connection attempts were observed with Wireshark in REMnux, when the infected machine (IP 192.168.180.130) sent SMB probe packets to a Windows host (IP 192.168.180.134), as shown in Fig. 5.

At the same time, the worm component attempts to spread across the external networks by generating various IP addresses and trying to connect to TCP port 445. This can be observed with Wireshark on REMnux, as shown in Fig. 6. The full list of WannaCry generated IP addresses obtained during our analysis is presented in Table VII.

During the SMB probing by WannaCry, one of the unique features of the generated traffic is that it contains two hardcoded IP addresses: 192.168.56.20 and 172.16.99.5. They can be observed by extracting strings from the binary. In particular, WannaCry sends three NetBIOS session setup packets, where two of them contain the aforementioned hadrcoded IP addresses.

During its execution, WannaCry also tries to contact the

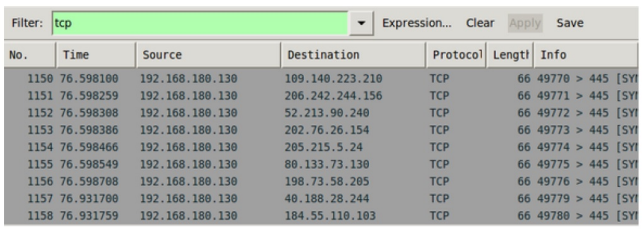

Fig. 6. WannaCry external network traffic attempting the SMB exploit.

TABLE VII. EXTERNAL IP ADDRESSES GENERATED BY WANNACRY.

IP address : port
109.140.223.210 : 445
206.242.244.156 : 445
52.213.90.240 : 445
202.76.26.154 : 445
205.215.5.24 : 445
80.133.73.130 : 445
198.73.58.205 : 445
40.188.28.244 : 445
184.55.110.103 : 445

C&C servers by parsing the contents of *c.wnry*, which specifies the configuration data, including the following *.onion* addresses to connect and the zipped Tor browser installation file:

$gx7ekbenv2riucmf.onion$
$57g7spgrzlojinas.onion$
$xxlvbrloxvriy2c5.onion$
$76jdd2ir2embyv47.onion$
$cwwnhwhlz52maqm7.onion$
$https://dist.torporject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip$

During its communication with Tor addresses, WannaCry establishes a secure HTTPS channel to port 443, and uses common Tor ports, 9001 and 9050, for network traffic and directory information.

VI. CONCLUSION AND FUTURE WORK

We have performed static and dynamic analysis of WannaCry ransomware. Both worm and encryption components of WannaCry have been examined using a wide range of reverse engineering and malware analysis tools. Our static analysis has revealed important information regarding the DLLs and the main Windows functions used by WannaCry, as well as about additional tools, such as the decryption component. Our dynamic analysis has revealed important characteristics and behaviours of WannaCry during its execution. In particular, we identified Tor addresses used for C&C, observed TCP and DNS connections, and SMB probes, as well as actions related to WannaCry persistence and obfuscation.

The findings of this work could be used for designing effective and efficient mitigation mechanisms for WannaCry and other ransomware families that exhibit similar behaviour. This is left as future work. In particular, we plan to investigate the use of software-defining networking (SDN) [13], [14] for ransomware detection and mitigation. SDN is an emerging paradigm of programmable networks, that decouples the control and data planes. SDN controllers maintain a view of the entire network and implement policy decisions. On the other hand, each device at the data plane maintains one or more *flow tables*, where the packet handling rules are stored. This changes the way that networks are designed and managed, and enables new SDN-based security solutions [15]–[17] such as firewalls and intrusion detection systems for various types of malware, including ransomware mitigation [18], [19].

REFERENCES

[1] D. O'Brien, "Ransomware 2017", Internet Security Threat Report, Symantec, July 2017.

[2] K. Savage, P. Coogan, and H. Lau, "The Evolution of Ransomware", Security Response, Symantec, June 2015.

[3] C. Everett, "Ransomware: To pay or not to pay?," Computer Fraud & Security, vol. 4, pp. 8-12, April 2016.

[4] McAfee Labs, "Understanding ransomware and strategies to defeat it," White Paper, 2016.

[5] Symantec, "What you need to know about the WannaCry ransomware," Threat Intelligence, October 2017.

[6] Microsoft Security Bulletin MS17-010 - Critical, March 14, 2017.

[7] REMnux: A Linux Toolkit for Reverse-Engineering and Analyzing Malware, https://remnux.org, accessed June 12, 2018.

[8] ViRus Share malware repository, https://virusshare.com, accessed June 12, 2018.

[9] Pestudio, Malware Assessment Tool, https://www.winitor.com, accessed June 12, 2018.

[10] IDA Pro, https://www.hex-rays.com/products/ida, accessed June 12, 2018.

[11] Tor Project, https://www.torproject.org, accessed June 12, 2018.

[12] WinHex: Computer Forensics and Data Recovery Software, https://www.x-ways.net/winhex, accessed June 12, 2018.

[13] B. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, future of programmable networks," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, Feb. 2014.

[14] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "A software-defined architecture for next-generation cellular networks," Proc. IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, May 2016.

[15] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," Computer Networks, vol. 85, pp. 1935, July 2015.

[16] J. M. Ceron, C. B. Margi, and L. Z. Granville, "MARS: An SDN-based malware analysis solution," Proc. IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, August 2016.

[17] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "On the security of software-defined next-generation cellular networks," Proc. IEICE Information and Communication Technology Forum (ICTF), Patras, Greece, July 2016.

[18] K. Cabaj and W. Mazurczyk, "Using software-defined networking for ransomware mitigation: The case of CryptoWall," IEEE Network, vol. 30, no. 6, pp. 14-20, Dec. 2016

[19] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics," Computer & Electrical Engineering, vol. 66, pp. 353-386, Feb. 2018.