Paper

WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms

Maxat Akbanov[1], Vassilios G. Vassilakis[2], and Michael D. Logothetis[3]

[1] *Department of Computer Science, University of York, York, United Kingdom*
[2] *University of York, York, United Kingdom*
[3] *WCL, Dept. of Electrical and Computer Engineering, University of Patras, Patras, Greece*

https://doi.org/10.26636/jtit.2019.130218

Abstract—In recent years, we have been experiencing fast proliferation of different types of ransomware targeting home users, companies and even critical telecommunications infrastructure elements. Modern day ransomware relies on sophisticated infection, persistence and recovery prevention mechanisms. Some recent examples that received significant attention include WannaCry, Petya and BadRabbit. To design and develop appropriate defense mechanisms, it is important to understand the characteristics and the behavior of different types of ransomware. Dynamic analysis techniques are typically used to achieve that purpose, where the malicious binaries are executed in a controlled environment and are then observed. In this work, the dynamic analysis results focusing on the infamous WannaCry ransomware are presented. In particular, WannaCry is examined, during its execution in a purpose-built virtual lab environment, in order to analyze its infection, persistence, recovery prevention and propagation mechanisms. The results obtained may be used for developing appropriate detection and defense solutions for WannaCry and other ransomware families that exhibit similar behaviors.

Keywords—dynamic malware analysis, ransomware, WannaCry.

1. Introduction

Ransomware threat is currently considered to be the main moneymaking scheme for cyber criminals and the key threat to Internet users [1], [2]. In recent years, the appearance of new types of ransomware has been observed, combining the use of worm-like spreading mechanisms and advanced recovery prevention schemes. Recent examples include WannaCry [3], [4] and Petya [5], [6], which exploit the weaknesses of Microsoft Windows, as well as BadRabbit [7], which spreads via insecure compromised websites.

From the defense perspective, the design of new countermeasures is considered, in addition to traditional security approaches, an important and trending task in this field. Such a design, however, requires a comprehensive analysis of ransomware functionality and behavior. This typically involves a wide range of malware analysis tools and techniques. Such techniques may be broadly classified as *static* and *dynamic*. Static analysis is performed without executing the malicious binary, while dynamic analysis involves executing the binary in an isolated environment.

In one of our previous works [8], we performed an initial static and dynamic analysis of WannaCry to identify its resources and functions, as well as its use of dynamic-link libraries (DLLs) and communication protocols. In this work, we have performed a comprehensive dynamic analysis, focusing on WannaCry's infection, persistence, recovery prevention and propagation mechanisms. The techniques presented are also applicable in the cases of other ransomware families whose characteristics are similar to that of WannaCry, such as worm-spreading mechanisms and public-key based encryption. In particular, the research presented examines WannaCry's behavior during its execution in a safe, purpose-built virtual lab environment at the University of York. The results obtained may form a basis for designing and developing effective ransomware defense solutions.

The rest of the paper is organized as follows. In Section 2, we present the relevant background information on ransomware in general and on WannaCry in particular. In Section 3, the main findings from the dynamic analysis of WannaCry we have performed, including its encryption process, recovery prevention and propagation mechanisms, are presented. Finally, Section 4 draws conclusions and discusses potential future directions.

2. Background

2.1. Ransomware

Ransomware is a type of malicious software (malware) that prevents users from accessing or limits their access to the system or files, either by locking the screen or by encrypting files, until a ransom is paid [9]. In most cases, ransomware leaves the user with very few options, such as only allowing the victim to communicate with the attacker and pay the ransom.

The most common types of ransomware use some form of encryption, including both symmetric and public-key based encryption schemes. Ransomware that relies on public-key encryption is particularly difficult to mitigate, since the encryption keys are stored in a remote command and control (C&C) server. There is usually a time limit for ransom to be paid, the users are provided 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 [10]: distribution, infection, C&C communications, file search, file encryption and ransom demand.

2.2. 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 [11]. According to multiple reports from security vendors, the total of 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.
The difficulty in protecting against WannaCry stems from its ability to spread 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 [12]. This vulnerability allows the adversaries to execute a remote code on the infected machines by sending specially crafted messages to an SMB v1 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 may 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 EternalBlue for 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.

3. WannaCry Analysis

In this section, we present our findings based on the dynamic analysis of WannaCry we have performed. Samples of WannaCry were obtained from VirusShare [13]. Two executable files were analyzed: the worm component and the encryption component (Table 1).

Table 1
WannaCry components

	Worm component
MD5	db349b97c37d22f5ea1d1841e3c89eb4
SHA1	e889544aff85ffaf8b0d0da705105dee7c 97fe26
SHA256	24d004a104d4d54034dbcffc2a4b19a11 f39008a575aa614ea04703480b1022c
File type	PE32 executable (GUI) Intel 80386, for MS Windows
	Encryption component
MD5	84c82835a5d21bbcf75a61706d8ab549
SHA1	5ff465afaabcbf0150d1a3ab2c2e74f3a4 426467
SHA256	ed01ebfbc9eb5bbea545af4d01bf5f1071 661840480439c6e5babe8e080e41aa
File type	PE32 executable (GUI) Intel 80386, for MS Windows

3.1. Testbed

In order to analyze WannaCry, a virtual testbed shown in Fig. 1 was built. The characteristics of the host machine are as follows: Intel Core i7-4700MQ 2.40 GHz and 16 GB RAM. The host machine acts as a virtual switch and is running REMnux [14], which is a free Linux toolkit for reverse engineering and malware analysis. Two virtual machines (VMs), running Windows 7 SP1, were used. The first VM was infected with WannaCry, whereas the other VM was clean. A custom network VMnet 5 – 192.168.180.0/24 was created with the Virtual Network Editor option in VMWare hypervisor. This testbed allows observing domain name system (DNS) queries made by WannaCry during the infection and replication process across internal and external

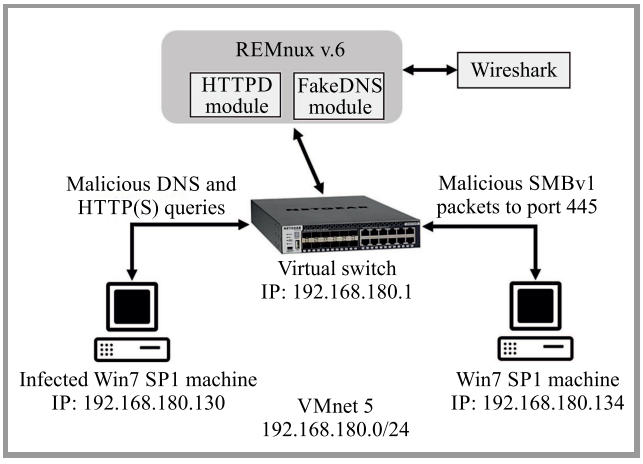

Fig. 1. Testbed for dynamic WannaCry analysis.

networks via port 445 of the SMB v1 protocol. The REMnux machine acts as a DNS and HTTP server, and is able to intercept all network communications using Wireshark. DNS and HTTP services in REMnux were enabled using FakeDNS and HTTP Daemon utilities, respectively.
The system level actions performed by WannaCry were observed on the infected Windows 7 SP1 machine with the 192.168.180.130 IP address. In order to observe and report the actions that WannaCry took while running on the system, the SysAnalyzer tool [15] was used. The main benefit of SysAnalyzer is that it is capable of taking system snapshots before and after malware execution, thus making it possible to inspect system attributes, such as running processes, open ports, DLLs loaded, registry key changes, run time file modifications, scheduled tasks, mutual exclusion objects (mutexes) and network connections. SysAnalyzer is also capable of taking memory dumps and scanning them for specific regular expressions. Before executing the WannaCry sample on the infected machine, the SysAnalyzer's configuration wizard was set to apply a 120 s delay between system snapshots, thus allowing to inspect all system attribute changes.

3.2. Libraries and Functions

Analysis performed with the Pestudio tool [16] revealed that the worm and the encryption components of WannaCry contain DLLs shown in Tables 2 and 3, respectively. During its execution, the worm component invokes *iphlpapi.dll* to retrieve network configuration settings for the infected host. *Kernel32.dll* and *msvcrt.dll* are the two libraries most frequently invoked by the encryption component. This may indicate that the main encryption functionality was implemented by these two malicious libraries. To confirm this, the imported functions of the libraries needed to be examined.

Table 2
DLLs of the worm component

Library	Imports	Description
ws2_32.dll	13	Windows Socket 2.0 32-bit DLL
iphlpapi.dll	2	IP Helper API
wininet.dll	3	Internet Extensions for Win32
kernel32.dll	32	Windows NT Base API Client DLL
advapi32.dll	11	Advanced Windows 32 Base API
msvcp60.dll	2	Windows NT C++ Runtime Library DLL
msvcrt.dll	28	Windows NT CRT DLL

Table 3
DLLs of the encryption component

Library	Imports	Description
kernel32.dll	54	Windows NT Base API Client DLL
advapi32.dll	10	Advanced Windows 32 Base API
user32.dll	1	Multi-User Windows User API Client DLL
msvcrt.dll	49	Windows NT CRT DLL

Table 4
Functions of the encryption component

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 5
Functions of the encryption component

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

The imported functions of the samples were observed by Pestudio. The most suspicious functions identified among them are shown in Tables 4 and 5. One may observe that in general, WannaCry uses Microsoft's crypto, file management and C runtime file APIs. The crypto API library is used to generate and manage random symmetric and asymmetric cryptographic keys.

```
root@remnux:~# fakedns 192.168.180.128
pyminifakeDNS:: dom.query. 60 IN A 192.168.180.128
Respuesta: watson.microsoft.com. -> 192.168.180.128
Respuesta: teredo.ipv6.microsoft.com. -> 192.168.180.128
Respuesta: www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com. -> 192.168.180.128
```

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

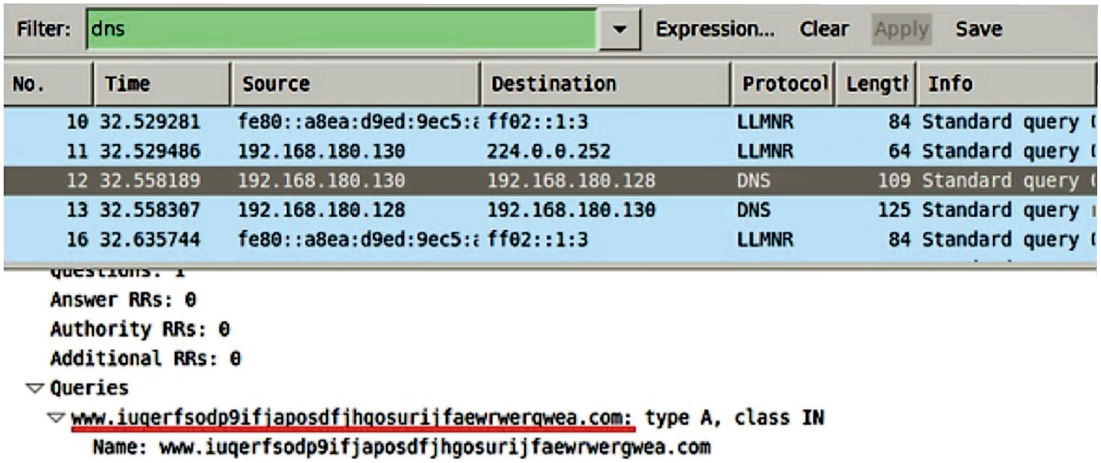

Fig. 3. Wireshark capture of the malicious DNS request.

3.3. Initial Interactions

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

www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

The aforementioned domain is a kill-switch domain. This means that 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" *mssecsvs2.0* process on the infected machine. Hence, this kill-switch domain may be used as part of a detection technique when developing a defense system.

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).

3.4. Persistence Mechanisms

After connection failure with the kill-switch domain, the worm component attempts to create a *mssecsvs2.0* process with the DisplayName of "Microsoft Security Center (2.0) Service". This can be observed in the Process Hacker tool with 4016 PID, indicating that the service has been launched (Fig. 4). In addition to this, the worm component of WannaCry extracts the hardcoded *R resource* binary and then copies it to "C:\Windows\taskche.exe" directory path. The R resource represents the binary of the WannaCry encryption component. After that, the worm runs the executable with the following parameters in the command line: "C:\Windows\taskche.exe/i". Next, the worm tries to move the "C:\Windows\taskche.exe" file to "C:\Windows\qeriuwjhrf", to replace the original file if it exists. This is done to ensure multiple infections and avoid any issues with creating the tasksche.exe process.

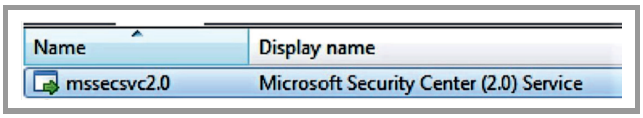

Fig. 4. Microsoft Security Center (2.0) Service.

Finally, WannaCry creates an entry in the Windows registry in order to ensure that it runs every time the computer is restarted. The new entry contains a string (e.g. "midtxzggq900"), which is a unique identifier randomly generated by using the computer name. Once the tasksche.exe component runs, it copies itself to a folder with a randomly generated name in the Common Appdata directory of the infected machine. Then, it attempts to establish memory persistence by adding itself to the AutoRun feature.

```
Created         C:\ProgramData\midtxzggq900\b.wnry
Modifed 15F936  C:\ProgramData\midtxzggq900\b.wnry
Created         C:\ProgramData\midtxzggq900\c.wnry
Modifed 30C     C:\ProgramData\midtxzggq900\c.wnry
Created         C:\ProgramData\midtxzggq900\msg
Created         C:\ProgramData\midtxzggq900\msg\m_bulgarian.wnry
Modifed         C:\ProgramData\midtxzggq900\msg
Modifed BB07    C:\ProgramData\midtxzggq900\msg\m_bulgarian.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_chinese (simplified).wnry
Modifed D457    C:\ProgramData\midtxzggq900\msg\m_chinese (simplified).wnry
Created         C:\ProgramData\midtxzggq900\msg\m_chinese (traditional).wnry
Modifed 135F2   C:\ProgramData\midtxzggq900\msg\m_chinese (traditional).wnry
Created         C:\ProgramData\midtxzggq900\msg\m_croatian.wnry
Modifed 989E    C:\ProgramData\midtxzggq900\msg\m_croatian.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_czech.wnry
Modifed 9E40    C:\ProgramData\midtxzggq900\msg\m_czech.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_danish.wnry
Modifed 90B5    C:\ProgramData\midtxzggq900\msg\m_danish.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_dutch.wnry
Modifed 907B    C:\ProgramData\midtxzggq900\msg\m_dutch.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_english.wnry
Modifed 906D    C:\ProgramData\midtxzggq900\msg\m_english.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_filipino.wnry
Modifed 92CC    C:\ProgramData\midtxzggq900\msg\m_filipino.wnry
Created         C:\ProgramData\midtxzggq900\msg\m_finnish.wnry
Modifed 95E9    C:\ProgramData\midtxzggq900\msg\m finnish.wnry
```

Fig. 5. WannaCry dropped files to the working directory.

Fig. 6. WannaCry extortion message.

In summary, the dynamic analysis has 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 the Windows command line,
- deletes all backup folders,
- by using the Windows command line, creates a VBScript program which generates a single shortcut of the *@WanaDecryptor@.exe* decrypter file,
- tries to kill SQL and MS Exchange database processes by executing several commands in the Windows command line.

3.5. Configuration Data Load

After the persistence phase, WannaCry loads the *XIA resource*, which corresponds to a password protected ZIP file. It decompresses the files and drops them to the working directory of the running process (Fig. 5), as observed in the DirWatch module of SysAnalyzer.
As one can see, WannaCry loads configuration data from the c.wnry file into memory. WannaCry randomly chooses one of the three available Bitcoin addresses and then writes this address back to the configuration data. This is done in order to display the payment address in the extortion message (Fig. 6). After that, WannaCry sets the hidden attribute (Fig. 7) for the working directory with the help of the CreateProcess function. Next, with the help of the Windows icacls command, WannaCry grants full access to all files on the target system (Fig. 8).

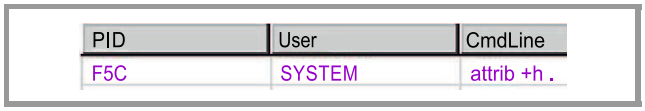

Fig. 7. WannaCry sets the hidden attribute for the working directory.

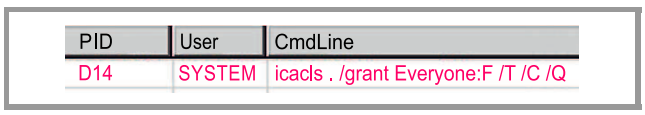

Fig. 8. WannaCry grants full access on the target system.

The next step is to import one of the hardcoded public RSA keys as was identified at offset 0xec00 of the tasksche.exe process (Fig. 9). WannaCry then loads and executes, in memory, the contents of the t.wnry file (Fig. 10) which contains the default encrypted AES key required for decrypting the DLL responsible for the file encryption routine. The first 8 bytes of the file are checked to match the WANACRY! string. Then, the imported public RSA key hardcoded within binary is used to decrypt the AES key stored at the beginning of the t.wnry file. The AES key obtained is then used to decrypt and load the encryption DLL, which can be observed with the help of OllyDbg debugging tool [17] during WannaCry execution, as shown in Fig. 11. This DLL is responsible for file encryption on the infected machine and is summarized in Table 6.

Table 6
Encryption DLL

MD5	f351e1fcca0c4ea05fc44d15a17f8b36
SHA1	7d36a6aa8cb6b504ee9213c200c831e b8d4ef26b
Size	65536 bytes
File type	Dynamic-Link-Library
Internal name	kbdlv.dll
File description	Latvia keyboard layout
Timestamp	Mon, Jul 13 18:12:55 2009

3.6. Encryption Process

The encryption component of WannaCry is invoked with the TaskStart system thread. During its execution, the encryption component checks if one of the following mutexes exists:

`GlobalnMsWinZonesCacheCounterMutexA`,
`GlobalnMsWinZonesCacheCounterMutexW`,
`MsWinZonesCacheCounterMutexA`.

If the mutex "MsWinZonesCacheCounterMutexA" is present, then the encryption component automatically stops without taking any further action. If the mutex is not present on the system, the encryption process starts. In particular, TaskStart creates a new mutex named "MsWinZonesCacheCounterMutexA" and reads the contents of the c.wnry file from the current directory. After that, WannaCry creates three configuration files shown in Table 7.

Table 7
WannaCry configuration files

Filename	Description
00000000.res	TOR/C2 info
00000000.pky	Public RSA key
00000000.eky	Encrypted private RSA key

After the configuration files have been created, the encryption component is ready to start encrypting files on the system. To accomplish this, it spawns several threads. First,

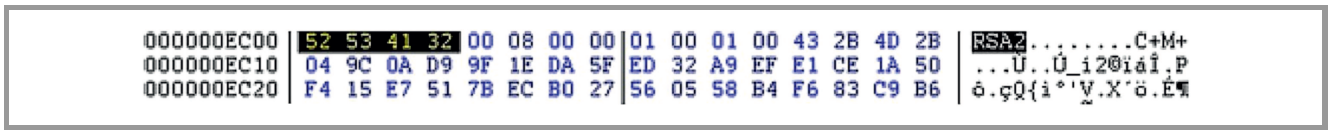

Fig. 9. Imported RSA private key.

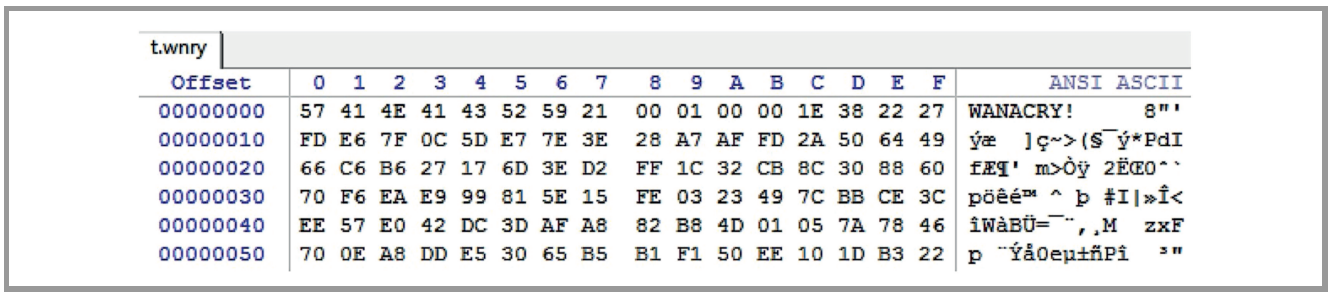

Fig. 10. Loaded and executed t.wnry file.

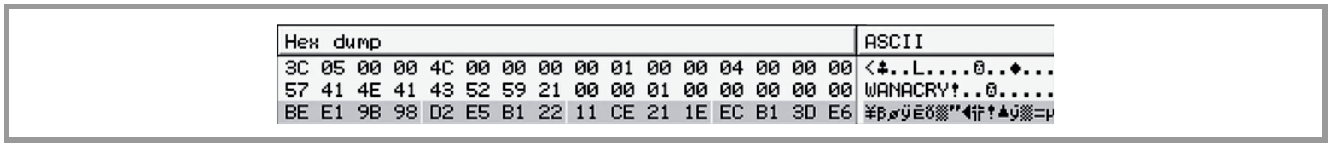

Fig. 11. Decrypted AES key in a memory dump.

WannaCry attempts to load and check the existence of two keys in the 00000000.pky and 00000000.dky files. The 00000000.dky file presents a decryption RSA key which is received upon the payment has been verified. When the victim clicks the "Check Payment" button, WannaCry starts checking for the presence of the 00000000.dky file on the system. If the two aforementioned files do not exist, WannaCry generates a new unique RSA 2048-bit asymmetric key pair, which can be seen in the memory dump made with with SysAnalyzer tool at 0x2B3795 offset (Fig. 12).

Offset	Data
2B3795	generating RSA key

Fig. 12. Generation of an RSA key pair.

Once the key pair has been generated, WannaCry exports the victim's public RSA key to a 00000000.pky file using Microsoft's *CryptExportKey* function. Next, WannaCry exports the victim's private RSA key and encrypts it with another hard-coded RSA public key. The encrypted private key is stored as a 00000000.eky file. After the key has been safely stored, WannaCry calls upon the *CryptDestroyKey* function to destroy the private key in memory, to limit any key recovery options.
Next, WannaCry starts enumerating, every 3 seconds, information about all logical drives attached to the system. If a new attached drive is not a CD ROM drive, then it begins the encryption process on the new drive. At this stage, WannaCry also starts iterating through all existing directories and searching for predefined file extensions of interest.

To encrypt each file, it generates a 16-byte symmetric AES key using the *CryptGenRandom* function. Then, it encrypts every generated AES key with the public RSA key and stores it inside the file header starting with the WANACRY! string value. Encrypted files are renamed and appended with the *.WNCRY* file extension.

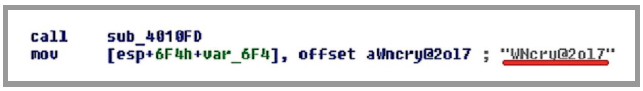

Fig. 13. Password for a ZIP archive in the encryption component.

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 [18] (see Fig. 13). The contents of the ZIP archive are summarized in Table 8 and described below:

- *msg* is a folder that contains a list of rich text format (RTF) files with the *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 the *.onion* extension and a link to a zipped installation file of the Tor browser from Tor Project [19];

Table 8
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

- *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 [20] by saving raw binary data with the .zip extension;
- *t.wnry* is an encrypted file with the WANACRY! encryption format. The file header starts with the WANACRY! string;
- *taskdl.exe* is a supporting tool for the deletion of files with the .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 the name of "@WanaDecryptor@.exe", which represents the decryption component of WannaCry.

At the same time, another thread calls the taskse.exe process every 30 s, which tries to enumerate active RDP sessions on connected remote machines and to run the @WanaDecryptor@.exe binary file. This file is extracted from the u.wnry file and represents the decryption component of WannaCry. The persistence of RDP session injections is ensured by adding the value in the AutoRun registry key.

3.7. Recovery Prevention

After finishing the encryption process, WannaCry tries to prevent various common data recovery methods by executing several commands on the system. To prevent data recovery, WannaCry executes the following commands:

- vssadmin delete shadows/all/quiet. Deletes all the shadow volumes on the system without alerting the user. By default, these volumes contain backup data in the event of a system fault;
- wmic shadowcopy delete. Ensures deletion of any copies relevant to shadow volumes;
- bcdedit/set default bootstatuspolicy ignoreallfailures. Ensures that the machine is booted, even if errors are found;
- bcdedit/set default recoveryenabled no. Disables the Windows recovery feature, thus preventing the victims from the possibility to reverting their system to a previous build;
- wbadmin delete catalog $-q$. Ensures that victim can no longer use any backup files created by Windows Server.

3.8. Propagation

The worm component of WannaCry carries the main propagation and exploit functionality, which utilizes the EternalBlue exploit and the DoublePulsar backdoor to leverage the MS17-010 SMB vulnerability [12]. After performing the initial interactions and checking connectivity with the kill-switch domain, the worm functionality is established by initiating the *mssecsvs2.0* service, which WannaCry installs after being executed. This service tries to spread WannaCry payload through the SMB vulnerability on any vulnerable systems on both internal and external networks.
In order to perform this, WannaCry creates and spawns two separate threads that simultaneously replicate worm payload in all detected networks. In the internal network, before starting the propagation process, the component obtains the IP addresses of local network interfaces by invoking the *GetAdaptersInfo* function, and determines the subnets existing in the network.
After that, the worm component tries to connect to all possible IP addresses in any available local network on port 445, which is the default port for SMB over IP service. If successful, the worm attempts to exploit the service for the MS17-010 vulnerability. In our testbed, connection attempts were observed with Wireshark on a REMnux machine, when the infected machine (IP 192.168.180.130) sent SMB probe packets to the clean machine (IP 192.168.180.134), as shown in Fig. 14.
During the SMB probing, 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 hardcoded IP addresses.
At the same time, the worm component attempts to spread across the external networks by generating various IP addresses and by trying to connect to TCP port 445. This can be observed with Wireshark on REMnux, as shown

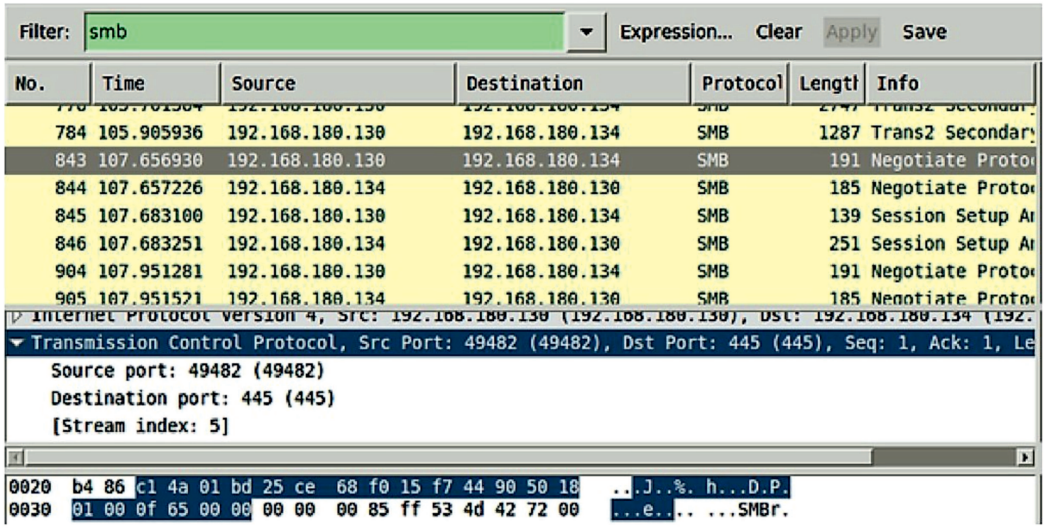

Fig. 14. WannaCry internal network traffic attempting the SMB exploit.

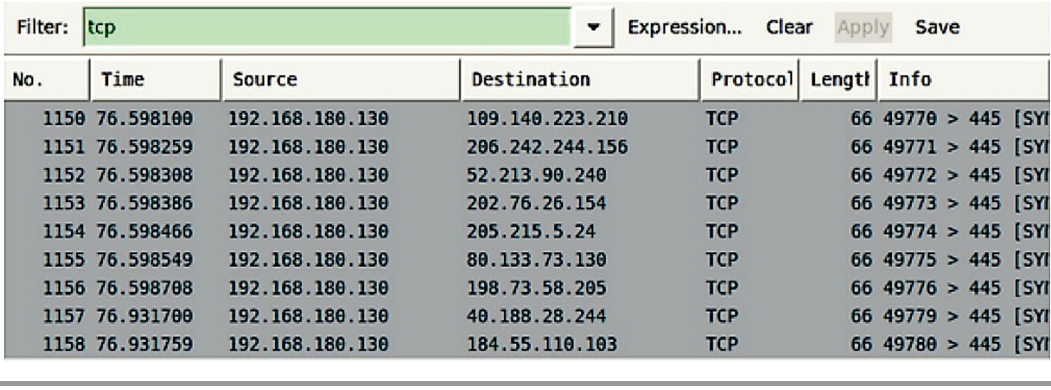

Fig. 15. WannaCry external network traffic attempting the SMB exploit.

in Fig. 15. As it can be seen, the worm attempts to probe external Internet IP addresses for the MS17-010 vulnerability. This explains the reason for the widespread infection seen during the massive outbreak on 12 May 2017. The full list of WannaCry generated IP addresses obtained during the analysis is presented in Table 9.

Table 9
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

3.9. C&C Communication

During its execution, the software also tries to contact the C&C servers. To this end, WannaCry unpacked and dropped files from the s.wnry file, containing the Tor executable, into the installation directory as shown

```
Created         C:\ProgramData\midtxzggq900\s.wnry
Modifed 2E5C4E  C:\ProgramData\midtxzggq900\s.wnry
```

Fig. 16. Tor executable dropped into the installation directory.

in Fig. 16. Before unpacking, it starts listening on the localhost address 127.0.0.1:9050. This address, with the specified 9050 port, is typically used for configuring the

Tor browser application. If the contents of the s.wnry file are corrupted, then WannaCry tries to download the Tor executable from a hardcoded URL. After the successful extraction of the Tor executable, it copies "TaskData\Tor\tor.exe" to "TaskData\Tor\taskhsvc.exe" and executes it. Next, WannaCry parses the contents of the c.wnry file, 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
```

After that, WannaCry sends the first eight bytes of the 00000000.res file content to the C&C server. These bytes specify the host and user name of the infected machine. The 00000000.res file, which is dropped during encryption process, accumulates in total 88 bytes of configuration data, including internal flags, counters, and timestamps.

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.

4. Conclusions and Future Work

We have performed a comprehensive dynamic analysis of WannaCry ransomware in a purpose-built virtual testbed. We analyzed the WannaCry version which was observed during the massive attacks on 12 May 2017. The analysis has revealed that the given ransomware is composed of two distinctive components, which enable the worm-like self-propagating mechanism and combined encryption process. Both worm and encryption components of WannaCry have been examined.

The focus of this study was on WannaCry's initial interactions and the infection process, its persistence mechanism, encryption process, recovery prevention as well as its propagation mechanisms and communication with C&C servers. The analysis has revealed important characteristics and behaviors of WannaCry during its execution. In particular, we identified Tor addresses used for C&C, observed TCP and DNS connections, SMB probes, as well as actions related to WannaCry persistence and obfuscation.

The worm component of WannaCry weaponized by the functionality enabling it to exploit and propagate via Microsoft's MS17-010 on unpatched systems by sending SMB probing packets on port 445. In addition to the modular nature of WannaCry, it was also observed that it has embedded RSA keys used to decrypt the required malicious DLL representing the encryption component. It was identified that the worm component scans both internal and external networks for MS17-010 vulnerability, by generating a list of local and global IP addresses. The worm tries to probe the hosts from the generated list by sending packets to port 445. Before its execution, WannaCry also performs an initial check with the kill-switch domain.

At the same time, the analysis has identified two hardcoded IP addresses (192.168.56.20 and 172.16.99.5), which are sent during the SMB probing. Depending on the condition of the s.wnry file dropped during execution, WannaCry can also communicate with embedded .onion addresses via a secure channel on port 443 and via common Tor ports 900 and 9050 to download the Tor browser installation software from a specified URL.

The findings of this work could be used for designing effective mitigation mechanisms for WannaCry and other ransomware families that exhibit similar behavior. This is left as future work. In particular, we plan to investigate the use of software-defining networking (SDN) [21], [22] 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 [23]–[25], such as firewalls and intrusion detection systems for various types of malware, including ransomware mitigation [26], [27].

References

[1] D. O'Brien, "Ransomware 2017", Internet Security Threat Report, Symantec, July 2017 [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-ransomware-2017-en.pdf

[2] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware", Security Response, Symantec, June 2015 [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf

[3] A. Zeichnick, "Self-propagating ransomware: What the WannaCry ransomworm means for you", May 2017 [Online]. Available: https://www.networkworld.com/article/3196993/security/self-propagating-ransomware-what-the-wannacry-ransomworm-means-for-you.html

[4] "Ransom.Wannacry", Symantec, May 2017 [Online]. Available: https://www.symantec.com/security-center/writeup/2017-051310-3522-99/

[5] "Petya – taking ransomware to the low level", Malwarebytes Labs, Jun. 2017 [Online]. Available: https://blog.malwarebytes.com/threat-analysis/2016/04/petya-ransomware/

[6] "Petya ransomware eats your hard drives", Kaspersky Labs, Jun. 2017 [Online]. Available: https://www.kaspersky.com/blog/petya-ransomware/11715

[7] "Bad Rabbit: A new ransomware epidemic is on the rise", Kaspersky Labs, Oct. 2017 [Online]. Available: https://www.kaspersky.com/blog/bad-rabbit-ransomware/19887/

[8] M. Akbanov, V. G. Vassilakis, I. D. Moscholios, and M. D. Logothetis, "Static and dynamic analysis of WannaCry ransmware", in *Proc. IEICE Inform. and Commun. Technol. Forum ICTF 2018*, Graz, Austria, 2018.

[9] C. Everett, "Ransomware: To pay or not to pay?", *Comp. Fraud & Secur.*, vol. 2016, no. 4, pp. 8–12, 2016 (doi: 10.1016/S1361-3723(16)30036-7).

[10] "Understanding ransomware and strategies to defeat it", McAfee Labs, White Paper, 2016 [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/white-papers/wp-understanding-ransomware-strategies-defeat.pdf

[11] "What you need to know about the WannaCry ransomware", Symantec, Threat Intelligence, Oct. 2017, [Online]. Available: https://www.symantec.com/blogs/threat-intelligence/wannacry-ransomware-attack

[12] Microsoft Security Bulletin MS17-010 – Critical, March 14, 2017 [Online]. Available: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010

[13] ViRus Share malware repository [Online]. Available: https://virusshare.com (accessed Nov. 30, 2018).

[14] "REMnux: A Linux toolkit for reverse-engineering and analyzing malware" [Online]. Available: https://remnux.org (accessed Nov. 30, 2018).

[15] SysAnalyzer – Automated malcode analysis system [Online]. Available: https://github.com/dzzie/SysAnalyzer (accessed Nov. 30, 2018).

[16] Pestudio, Malware Assessment Tool [Online]. Available: https://www.winitor.com (accessed Nov. 30, 2018).

[17] OllyDbg – A 32-bit assembler level debugger for Microsoft Windows [Online]. Available: http://www.ollydbg.de/ (accessed Nov. 30, 2018).

[18] IDA: Pro [Online]. Available: https://www.hex-rays.com/products/ida (accessed Nov. 30, 2018).

[19] Tor Project [Online]. Available: https://www.torproject.org (accessed Nov. 30, 2018).

[20] "WinHex: Computer forensics and data recovery software" [Online]. Available: https://www.x-ways.net/winhex (accessed Nov. 30, 2018).

[21] 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 Commun. Surveys & Tutor.*, vol. 16, no. 3, pp. 1617-1634, 2014 (doi: 10.1109/SURV.2014.012214.00180).

[22] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "A software-defined architecture for next-generation cellular networks", in *Proc. IEEE Int. Conf. on Commun. ICC 2016*, Kuala Lumpur, Malaysia, 2016 (doi: 10.1109/ICC.2016.7511018).

[23] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study", *Comp. Netw.*, vol. 85, pp. 19–35, 2015 (doi: 10.1016/j.comnet.2015.05.005).

[24] J. M. Ceron, C. B. Margi, and L. Z. Granville, "MARS: An SDN-based malware analysis solution", *Proc. IEEE Symp. on Comp. and Commun. ISCC 2016*, Messina, Italy, 2016 (doi: 10.1109/ISCC.2016.7543792).

[25] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "On the security of software-defined next-generation cellular networks", in *Proc. IEICE Inform. and Commun. Technol. Forum ICTF 2016*, Patras, Greece, 2016.

[26] 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, 2016 (doi: 10.1109/MNET.2016.1600110NM).

[27] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics", *Comp. & Elec. Engin.*, vol. 66, pp. 353–386, 2018 (doi: 10.1016/j.compeleceng.2017.10.012).

Maxat Akbanov received the B.Sc. degree in Information and Communications System Security from the National Technical University of Ukraine "Kyiv Polytechnic University", Kyiv, Ukraine, in 2011, and the M.Sc. degree in Cyber Security from the University of York, York, UK, in 2018. In 2008 and 2016, he received the prestigious Kazakhstan governmental "Bolashak" scholarship to fund his studies abroad. He holds merit and distinction awards for B.Sc. and M.Sc. degrees, respectively. He is currently working for the private sector in Kazakhstan and is involved in developing several startup projects for the government-sponsored "Digital Kazakhstan" and "Cyber Shield" strategies. His main research interests include network and malware forensics, software-defined networking, covert channels, cryptography, Internet of Things, machine learning and artificial intelligence.
E-mail: maxat.akbanov@gmail.com
Department of Computer Science
University of York
Deramore Lane
Heslington
York YO10 5GH, United Kingdom

Vassilios G. Vassilakis received his Ph.D. degree in Electrical and Computer Engineering from the University of Patras, Greece in 2011. He is currently a lecturer in Cyber Security at the University of York, UK. He's been involved in EU, UK, and industry funded R&D projects related to the design and analysis of future mobile networks and Internet technologies. His main research interests are in the areas of network security, Internet of Things, next-generation wireless and mobile networks, and software-defined networks. He has published over 90 papers in international journals/conferences. He has served as a Guest Editor in IEICE Transactions on Communications, IET Networks, and Elsevier Optical Switching & Networking, and in the TPC of IEEE ICC and IEEE Globecom.
E-mail: vv274@cl.cam.ac.uk
University of York
York YO10 5DD, United Kingdom

Michael D. Logothetis received his B.Eng. degree and Ph.D. in Electrical Engineering, both from the University of Patras, Patras, Greece, in 1981 and 1990, respectively. From 1991 to 1992 he was a Research Associate at NTT's Telecommunication Networks Laboratories, Tokyo, Japan. In 2009 he was elected (Full) Professor at the ECE Department of the University of Patras. His research interests include teletraffic theory, simulation and performance optimization of telecommunications networks. He has published over 200 conference/journal papers. He has become a Guest Editor in: Mediterranean Journal of Electronics and Communications, Mediterranean Journal of Computers and Networks, IET Circuits, Devices and Systems, IET Networks and Ubiquitous Computing and Communication Journal. He is a member of the IARIA (Fellow), IEEE (Senior), IEICE (Senior), FITCE and the Technical Chamber of Greece (TEE).

E-mail: mlogo@upatras.gr
Wire Communications Laboratory
Department of Electrical and Computer Engineering
University of Patras
265 04 Patras, Greece